Institut für Parallele und Verteilte Systeme
Abteilung Simulation großer Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. 215

Echtzeit-Festkörper-Simulation mit der Lattice Boltzmann Methode

Axel Reiser

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. rer. nat. habil. Miriam Mehl
Betreuer/in:	M.Sc. Fabian Franzelin, Dipl.-Inf. Michael Lahnert
Beginn am:	8. Dezember 2014
Beendet am:	9. Juni 2015
CR-Nummer:	I.6.4

Kurzfassung

Die Lattice Boltzmann Methode ist ein beliebtes Verfahren zur Simulation von Strömungen v.a. wegen der schnellen Berechnung. Bei diesem mesoskopischen Modell wird die Fluidbewegung mithilfe von probabilistischen Verteilungen erzeugt. Diese werden auf dem Gitter an umliegende Punkte übertragen, was dann den Fluss bildet. Die Verteilung werden über die Berücksichtigung der Partikelkollisionen bestimmt.

Für die Kopplung mit der Festkörper-Simulation müssen verschiedene Probleme gelöst werden. So muss diese in den Programmablauf integriert werden und die Festkörper in das Gitter der Lattice Boltzmann Methode. Dann müssen neue Randbedingungen definiert werden, da die Festkörper nicht diskretisiert sind. Eine Folge der Kopplung ist auch, dass nun Initialisierungsverfahren für Zellen benötigt werden. Zuletzt müssen noch die Festkörper vom Fluid angetrieben werden.

Die Kopplungsverfahren werden analysiert. Dabei wird besonderen Wert auf die Echtzeitfähigkeit gelegt. Es sollen sich sowohl die Lattice Boltzmann Simulation, als auch die Festkörper richtig verhalten. Dafür wird überprüft, ob die Erhaltungssätze der Lattice Boltzmann Methode noch gelten, und das Strömungsbild untersucht. Bei den Festkörpern wird auf richtige Beschleunigung und Rotation getestet. Für den Vergleich der Verfahren werden die Unterschiede nicht analytisch, sondern optisch bestimmt, da der Benutzer die Simulationsergebnisse in Echtzeit visuell wahrnimmt. Dazu gehört auch die Untersuchung der Laufzeit.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Algorithmen

1. Einleitung

Strömungssimulation wird in vielen Felder der Forschung genutzt. So können z.B. die aerodynamischen Eigenschaften von Fahrzeugen oder das Verhalten eines Feuers in Brandszenarien (siehe [Pfi12]) analysiert werden. Es werden aber auch in der Computergrafik immer häufiger realistische Simulationen verlangt. Bei den Special Effects in der Filmindustrie wird sehr hohen Wert auf Realismus gelegt und es werden deshalb aufwendige Simulationen verwendet, auch für Strömungen. Mit der immer weiter zunehmenden Leistung der Heim-PCs wird auch in der Spieleindustrie bei Computerspielen verstärkt auf das Simulieren von physikalischen Vorgängen gesetzt. Um die Strömungssimulation auch effektiv nutzen zu können, muss sie ausreichend adaptiv sein, damit sie sich dem für den Nutzer interessanten Anwendungsfall anpassen kann. Dazu gehört auch die in dieser Arbeit behandelte Kopplung von Strömungssimulation und Festkörpern.

Für die Simulation von Strömungen gibt es verschiedene Modelle. Diese kann man in drei Kategorien unterteilen:

- makroskopische Modelle
- mikroskopische Modelle
- mesoskopische Modelle

Makroskopische Modelle betrachten das Fluid als eine Einheit. Sie berechnen die Fluideigenschaften ohne Rücksicht darauf zu nehmen, dass das Fluid eigentlich aus Partikeln besteht. Diese Modelle werden durch partielle Differenzialgleichungen beschrieben. Makroskopische Modelle bei Strömungen sind die Navier-Stokes-Gleichungen und deren Erweiterungen. Genaueres zu diesen erfährt man in [Con88].

Im Gegensatz zu den makroskopischen Modellen ignorieren die mikroskopischen das Gesamtbild und betrachten stattdessen die einzelnen Partikel des Fluids. Dabei werden die Wechselwirkungen zwischen den Fluidpartikeln und deren resultierende Bewegung berechnet. Dadurch erhält man ein Strömungsbild auf dem Gebiet. Zu den mikroskopischen Modellen zählt die Molekulardynamik, welche in [Rap04] erläutert wird.

Die mesoskopischen Modelle bilden einen Mittelweg zwischen den makroskopischen und mikroskopischen. Sie haben eine grobkörnige Darstellung des Fluids ähnlich wie bei den makroskopischen Modellen, berücksichtigen aber noch die Eigenschaften der Partikelkollisionen, wie es bei den mikroskopischen Modellen üblich ist. Zu diesen mesoskopischen Modellen zählt die Lattice Boltzmann Methode (LBM).

Bei den Modellen wird weiterhin noch zwischen Lagrange'schen und Euler'schen Methoden unterschieden.

Lagrange'sche Methoden betrachten jeweils die einzelnen Partikel des Fluids und verfolgen diese im Gebiet. Dabei können die Partikel auch aus großen Atom- bzw. Molekül-Gruppen bestehen. So teilt die Smoothed Particles Hydrodynamics (SPH) Methode das Fluid in Partikel und approximiert in jedem Zeitschritt deren Eigenschaften mithilfe von Integralinterpolation. [Mon92] liefert einen guten Einstieg für weitere Recherchen bezüglich SPH.

Bei Euler'schen Methoden hingegen hat man mehrere ortsfeste Beobachter, die die Eigenschaften des Fluids an einer bestimmten Stelle auf dem Gebiet bestimmen. Bringt man diese Beobachter in uniformen Abständen auf dem Gebiet an, erhält man ein Gitter. Daher arbeiten Euler'sche Methoden meist auf Gittern. Dazu gehört auch die Lattice Boltzmann Methode. Diese berechnet verschiedene Eigenschaften des Fluids, wie z.B. die Dichte oder die Strömungsgeschwindigkeit, für jeden Gitterpunkt.

In dieser Arbeit wird die Lattice Boltzmann Methode verwendet, da diese u.a. aufgrund des lokalen Operators sowohl einfach zu implementieren, als auch schnell zu berechnen ist. Damit wird das Erreichen einer Simulation in Echtzeit ein realistisches Ziel.

Gliederung

Die Arbeit ist in folgender Weise gegliedert:

Kapitel 2 – LBM Grundlagen: Hier werden die Grundlagen der Lattice Boltzmann Methode beschrieben.

Kapitel 3 – Kopplungsmethoden und Implementierung stellt die verwendeten Kopplungsmethoden vor und liefert Hinweise für die Implementierung.

Kapitel 4 – Anwendung und Analyse zeigt eine Anwendung und die Analyse der vorgestellten Methoden.

Kapitel 5 – Fazit und Ausblick fasst die Ergebnisse der Arbeit zusammen und liefert einen Ausblick.

2. LBM Grundlagen

Die Lattice Boltzmann Methode ist ein Verfahren zur Simulation von Strömungen auf einem Gitter. Sie arbeitet auf mesoskopischer Skala und somit mit Populationen auf den Gitterpunkten. Verteilungen beschreiben den probabilistischen Übergang von Fluid auf die umliegenden Punkte. Die Berechnungen dieser Verteilungen beziehen die Eigenschaften der Partikelkollisionen mit ein. Ein Zeitschritt der Simulation besteht aus dem Berechnen und darauf folgenden Übergeben der Verteilungen an die angrenzenden Gitterpunkte. Auf dem Rand des Gebietes werden Randbedingungen definiert.

Die Korrektheit des Verfahrens wird mithilfe der Chapman-Enskog-Erweiterung bewiesen. Diese zeigt die Approximation der Navier-Stokes-Gleichungen.

2.1. Die Lattice Boltzmann Methode

Bei der Lattice Boltzmann Methode handelt es sich um einen zellulären Automaten. Ein solcher wird mit folgenden Größen beschrieben:

- ein Raum/Gebiet Ω
- eine Nachbarschaft N
- eine Zustandsmenge $\mathbb{L}$
- eine Zustandsübergangsfunktion $\delta : \mathbb{L}^N \to \mathbb{L}$

Für das Gebiet Ω wird das LBM-Gitter verwendet. Eine Nachbarschaft N muss gewählt werden. Für den 2D-Fall wird hier das D2Q9-Modell verwendet. Dieses wird in Abbildung 2.1 dargestellt. Dabei bewegen sich die Fluidpartikel mit einer diskreten Geschwindigkeit $\mathbf{c}_i$ in i-Richtung ($i \in \{0, ..., 8\}$). Die Partikel können sich in einem Zeitschritt nur um eine Zelle weiterbewegen, d.h. es gibt ein festes $\Delta\mathbf{x}$, das in jeder Richtung jeweils den Abstand zum nächsten Gitterpunkt beträgt. Das führt in Verbindung mit den diskreten Geschwindigkeiten $\mathbf{c}_i$ zu einem festes Δt, also einen festen Zeitschritt. Das hat den Vorteil, dass man nicht extra die obere Schranke des Zeitschrittes bestimmen muss, wie z.B. bei makroskopischen Modellen auf Basis der Navier-Stokes-Gleichungen.

Auf den Gitterpunkten der Lattice Boltzmann Methode sind Populationen von Fluidpartikeln vorhanden. Da deren Werte kontinuierliche, reelle Zahlen sind, setzt man die Zustandsmenge $\mathbb{L} = \mathbb{R}$. Unter der Annahme von molekularem Chaos kann die Zustandsübergangsfunktion δ, die die Verteilung dieser Populationen in jedem Zeitschritt bestimmt, mithilfe der Boltzmann-Gleichung berechnet

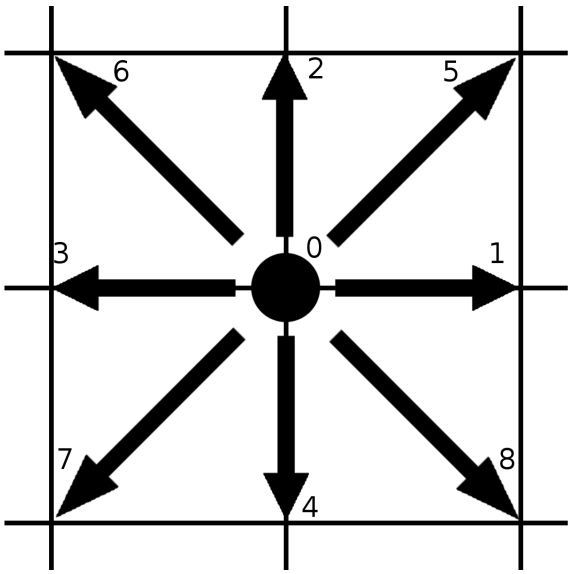


Abbildung 2.1.: Darstellung des D2Q9-Modells auf einem Gitterpunkt mit den dazugehörigen Geschwindigkeiten. Die Nummerierung der Pfeile entspricht der Richtung i. $c = \frac{\Delta x}{\Delta t}$ wird zum Beginn der Simulation festgelegt.

werden. Molekulares Chaos geht davon aus, dass die Geschwindigkeiten von kollidierenden Fluidpartikeln unabhängig sind von der Position und nicht korrelieren. Hier wird wie in [Lad94] ein diskretes Gegenstück der Boltzmann-Gleichung verwendet:

$$f_i(\mathbf{x} + \mathbf{c}_i\Delta t, t + \Delta t) - f_i(\mathbf{x}, t) = \quad \Delta_i(f) \tag{2.1}$$

$\Delta_i(f)$ ist dabei der Kollisionsterm. f_i mit $i = 0, ..., 8$ sind die Verteilungen in der entsprechenden i-Richtung. Die Boltzmann-Gleichung wird in zwei Schritte geteilt.

$$f_i^*(\mathbf{x}, t) = \quad f_i(\mathbf{x}, t) + \Delta_i(f) \tag{2.2}$$

$$f_i(\mathbf{x} + \mathbf{c}_i\Delta t, t + \Delta t) = \qquad f_i^*(\mathbf{x}, t) \tag{2.3}$$

(2.2) wird Kollisionschritt, (2.3) Strömungsschritt genannt. Bei diesem Zwei-Schritt-Verfahren handelt es sich um ein Push Schema, da die Verteilungen direkt nach der Kollision propagiert werden. Dieser Vorgang wird in Abbildung 2.2 verdeutlicht. Im Gegensatz dazu werden bei der Pull-Variante die Verteilungen am Anfang eines Zeitschrittes aus den umliegenden Zellen ausgelesen. Die verschiedenen Propagierungsschritte werden in [WZHW94] erläutert.

Als Kollisionsterm $\Delta_i(f)$ wird hier der BGK-Operator verwendet:

$$\Delta_i(f) = -\frac{1}{\tau}(f_i - f_i^{eq}) \tag{2.4}$$

τ ist die Relaxationszeit des Gitters. Diese bestimmt wie schnell sich das Fluid an das Gleichgewicht annähert und hat direkten Einfluss auf die Viskosität ν des Fluids.

$$\nu = c_s^2 dt(\tau - 0.5) \tag{2.5}$$

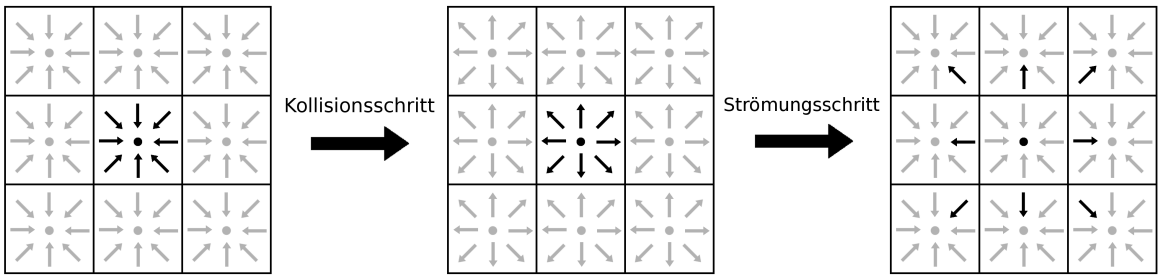


Abbildung 2.2.: Darstellung der Verteilungen über einen Zeitschritt hinweg für eine Zelle mit ihren Nachbarn.

Dabei ist $c_s = \frac{1}{\sqrt{3}}$ die Lattice-Schallgeschwindigkeit.

f_i^{eq} ist die Equilibriumsfunktion. Sie approximiert mithilfe einer Taylor-Entwicklung die Maxwell-Boltzmann-Gleichgewichtsverteilung $f^{(0)}$ und damit die stationäre Lösung der Boltzmann-Gleichung, also das lokale Gleichgewicht.

$$f_i^{eq}(\mathbf{x},t) = f_i^{eq}(\rho(\mathbf{x},t),\mathbf{u}(\mathbf{x},t)) = w_i\rho\left[1 + c_s^{-2}\mathbf{c}_i\cdot\mathbf{u} + \frac{c_s^{-4}}{2}(\mathbf{c}_i\cdot\mathbf{u})^2 - \frac{c_s^{-2}}{2}\mathbf{u}^2\right] \tag{2.6}$$

Als Parameter werden Dichte und Flussgeschwindigkeit an einem Gitterpunkt an der Stelle $\mathbf{x}$ benötigt. Diese beiden Werte werden aus den einkommenden Verteilungen berechnet.

$$\rho(\mathbf{x},t) = \sum_{i=0}^{8} f_i \tag{2.7}$$

$$\mathbf{u}(\mathbf{x},t) = \frac{1}{\rho(\mathbf{x},t)}\sum_{i=0}^{8} f_i\mathbf{c}_i \tag{2.8}$$

Die Gewichte w_i für das D2Q9-Modell sind wie folgt:

$$w_0 = \frac{4}{9} \tag{2.9}$$

$$w_{1,2,3,4} = \frac{1}{9} \tag{2.10}$$

$$w_{5,6,7,8} = \frac{1}{36} \tag{2.11}$$

Eine Gewichtung ist notwendig, da sich die Längen der diskreten Geschwindigkeiten $\mathbf{c}_i$ unterscheiden und berücksichtigt werden müssen.

Damit ein LBM-Modell korrekt funktioniert, müssen die Erhaltungssätze der Navier-Stokes-Gleichungen gelten. Das sind Massenerhaltung, Impulserhaltung und Energieerhaltung. Eine Anwendung dieser Sätze findet sich in [Sch08]. Darin wird u.a. die Massenerhaltung und Impulserhaltung definiert.

Für die Massenerhaltung muss gelten:

$$\left(\sum_i f_i^{eq}\right) - \rho = 0 \tag{2.12}$$

$$\sum_i \Delta_i(f) = 0 \tag{2.13}$$

Für die Impulserhaltung muss gelten:

$$\left(\sum_i f_i^{eq}\mathbf{c}_i\right) - \rho\mathbf{u} = \mathbf{0} \tag{2.14}$$

$$\sum_i \Delta_i\mathbf{c}_i = \mathbf{0} \tag{2.15}$$

Für eine ausführlichere Einführung der LBM-Grundlagen werden [Suc01] und [WG00] empfohlen.

2.2. Randbedingungen

Hier werden vier Arten von Randbedingungen vorgestellt. Periodische Randbedingungen stellen sinnvolle Ergebnisse sicher, auch wenn das Feld nicht durch Wände begrenzt ist, no-slip bounce-back Randbedingungen beschreiben die Abstoßung von einer Wand. Zusätzlich werden auch noch Ein- und Ausfluss als Randbedingung definiert.

Im Allgemeinen gibt es ein Gebiet Ω. Dieses Gebiet hat den Rand $\partial\Omega$. Das Verhalten von $f_i(\mathbf{x} + \mathbf{c}_i\Delta t, t + \Delta t)$ ist dabei nicht vollständig spezifiziert, wenn $\mathbf{x} + \mathbf{c}_i\Delta t \in \partial\Omega$ oder wenn $\mathbf{x} \in \partial\Omega$ und $\mathbf{x} + \mathbf{c}_i\Delta t \in \Omega$. Das sind die Verteilungen, die vom Gebiet in den Rand führen, und die, die vom Rand in das Gebiet zurückkommen. Randbedingungen müssen diese Verteilungen definieren, damit die LBM mit dem Rand umgehen kann.

2.2.1. Periodische Randbedingungen

Die einfachste Methode ist die der periodischen Randbedingungen. Dabei werden Verteilungen, die auf den Rand treffen, auf die andere Seite des Gebiets übertragen. Dies erreicht man, indem man für jede Richtung (außer $i = 0$) an der jeweiligen Seite des Rands eine Replikation des Gebietes legt. Bei dem D2Q9-Modell legt man also das Feld acht mal um sich herum und hat dann einen neun Felder Würfel. Da die LBM lediglich mit den benachbarten Gitterzellen arbeitet, benötigt man nur eine Zellschicht der Kopien um das eigentliche Gebiet herum. Diese wird in Abbildung 2.3 veranschaulicht.

Während des Strömungsschrittes dienen die Replikationen als Referenz auf das Feld. Die Verteilungen werden dann in die passende Gitterzelle geschrieben.

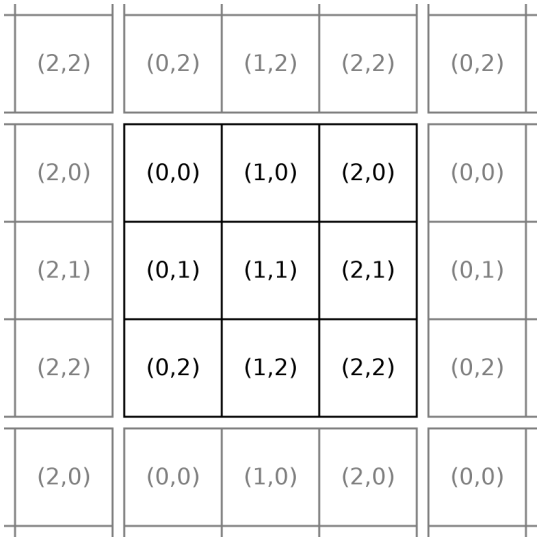


Abbildung 2.3.: Imaginäre Zellschicht um das Gitter, die zeigt, wohin Verteilungen gestreamt werden.

2.2.2. No-slip bounce-back Randbedingungen

Befindet sich am Rand des Fluids eine feste Wand, werden im Rahmen der vorliegenden Arbeit no-slip bounce-back Randbedingungen verwendet. Wände werden in das LBM-Gitter integriert, indem dry-nodes und wet-nodes definiert werden. Dry-nodes sind Gitterpunkte, die zur Wand zählen, wet-nodes sind Fluidgitterpunkte. So können auch komplexe Geometrien verwendet werden, indem man passende Gitterpunkte in dry-nodes ändert. Um die von der Wand reflektierten Verteilungen zu bestimmen, muss man festlegen, wo sich der Rand der Wand, also die Wand-Fluid-Grenze, in der Simulation befindet. Dabei werden zwei Fälle unterschieden: Die Grenze liegt genau auf den dry-nodes, on-grid genannt, oder genau zwischen dry-nodes und wet-nodes, also mid-grid. Hier wurde die on-grid Variante benutzt. Diese reflektiert die ausgehenden Verteilungen auf den gleichen Fluidpunkt zurück. Abbildung 2.4 veranschaulicht dieses Verfahren.

Rechnerisch erhält man die einkommenden Verteilungen an einem Fluidpunkt vom Rand wie folgt:

$$\begin{aligned} f_4(\mathbf{x}, t+\Delta t) &= f_2^*(\mathbf{x}, t) \\ f_7(\mathbf{x}, t+\Delta t) &= f_5^*(\mathbf{x}, t) \\ f_8(\mathbf{x}, t+\Delta t) &= f_6^*(\mathbf{x}, t) \end{aligned} \tag{2.16}$$

Diese Methode funktioniert nur für statische Wände, kann aber auf bewegliche wie folgt erweitert werden: Hat man einen beweglichen Rand, muss man diese Bewegung in die reflektierten Verteilungen miteinberechnen. Dafür wird in die Gleichung der no-slip Randbedingungen ein Geschwindigkeitsterm eingefügt, wie in [Neu13] vorgestellt.

$$f_{\bar{i}}(\mathbf{x}, t+dt) = f_i^*(\mathbf{x}, t) - \frac{2}{c_s^2} w_i \rho_w (\mathbf{c}_i \cdot \mathbf{v}_w) \qquad \text{mit } \bar{i}\text{, sodass } \mathbf{c}_{\bar{i}} = -\mathbf{c}_i \tag{2.17}$$

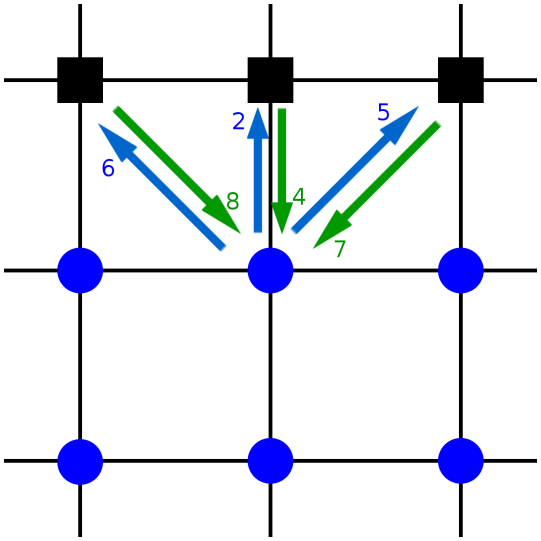


Abbildung 2.4.: Verteilungen bei on-grid no-slip bounce-back Randbedingungen. Blau sind wet-nodes bzw. vom Fluid ausgehende Verteilungen, schwarz sind dry-nodes und grün sind von der Wand reflektierte Verteilungen. Die Zahlen entsprechen der Richtung i.

$\rho_w = \rho(\mathbf{x}, t)$ ist dabei die Dichte in der aktuellen Zelle, $\mathbf{v}_w$ die Geschwindigkeit, mit der sich die Wand bewegt. Trotz der Bewegung des Randes geht man davon aus, dass dieser sich in jedem Zeitschritt on-grid befindet.

2.2.3. Einfluss

Der Einfluss sorgt dafür, dass Fluid in das Gebiet einströmt. Da dieser Strom unabhängig vom bereits vorhanden Fluid im Gebiet ist, können die einkommenden Verteilungen ignoriert werden. Die ausgehenden Verteilungen werden auf einen meist konstanten Wert gesetzt, der dann die Geschwindigkeit des Fluids in der Strömung bestimmt. Dafür bieten sich die Equilibriumsverteilungen an. Diese ermöglichen das Erzeugen einer Strömung mit einer bestimmten Geschwindigkeit $\mathbf{u}_E$ und Dichte ρ_E.

$$f_i = f_i^{eq}(\mathbf{u}_E, \rho_E) \tag{2.18}$$

Die Equilibriumsverteilungen werden dabei wie in Gleichung 2.6 berechnet.

Zusätzlich gibt es auch ungerichtete Einströmungen. Bei diesen „schwappt“ das Fluid sozusagen gleichmäßig in das Gebiet über. Dabei kann man die ausgehenden Verteilungen mit einer Konstanten c_E bestimmen.

$$f_i = w_i c_E \tag{2.19}$$

Poiseuille-Fluss Ein Poiseuille-Fluss ist ein spezifisches Strömungsbild, das sich unter bestimmten Voraussetzungen in einem Kanal bilden kann. Ein Kanal ist dabei definiert als ein Gebiet, das an den Rändern durch eine feste Wand begrenzt ist, an einer Seite einen Einfluss und an der gegenüberliegenden einen Ausfluss hat. Um einen Poiseuille-Fluss zu erhalten, muss die Geschwindigkeit des einfließenden Fluids geringer sein, je näher der Gitterpunkt des Einflusses zum Rand ist. Das führt zu einem parabelförmigen Geschwindigkeitsprofil (siehe Abbildung 2.5). Die folgende Vorschrift aus

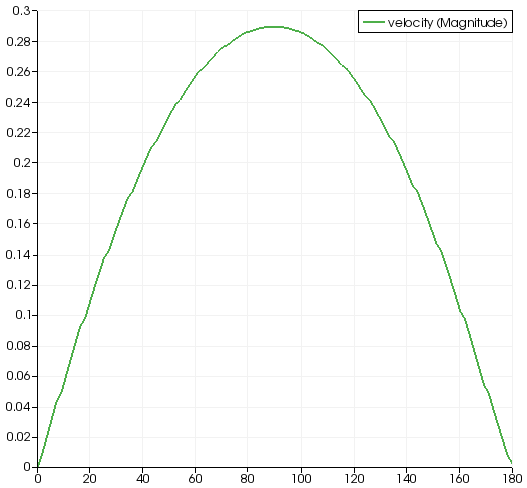


Abbildung 2.5.: Geschwindigkeitsprofil eines Querschnitts der Strömung eines Poiseuille-Flusses.

[ST96] erzeugt ein solches:

$$u(0, y) = 4u_{\max}\frac{y(H-y)}{H^2} \tag{2.20}$$

H ist die Höhe des Kanals, die längs zur Einfluss-Fluid-Grenze verläuft, und $u_{\max}$ die maximale Geschwindigkeit. Die feste Wand zählt bei dieser Berechnung nicht zum Kanal. Die berechneten Geschwindigkeiten können dann zur Bestimmung der Verteilungen in Gleichung (2.18) eingesetzt werden.

2.2.4. Ausfluss

Um einen gleichmäßigen Fluss zu erhalten, darf nur genau so viel Fluid das System am Ausfluss verlassen, wie ihm am Einfluss zugeführt wird. Dies wird mit porous-plug Randbedingungen erreicht. Dabei wird ein Teil des Fluids das an den Ausfluss kommt wieder zurück reflektiert. Der Anteil wird so gewählt, dass die Masseerhaltung gesichert ist. Dafür wird in [Suc01] ein r definiert mit:

$$r = 1 + 4\frac{u - u_{in}}{1 - 2u} \tag{2.21}$$

u ist die Geschwindigkeit des Fluids am Ausfluss, u_{in} die Einflussgeschwindigkeit. Wenn $u = u_{in}$, müssen keine Verteilungen reflektiert werden und $r = 1$. Ist $u > u_{in}$, müssen Verteilungen über-

reflektiert werden ($r > 1$). Dementsprechend werden bei $r < 1$ die reflektierten Verteilungen so gesetzt, dass die Strömung zum Ausfluss beschleunigt.

Unter der Annahme, dass u und u_{in} konstant ist, hat man ein konstantes r und damit konstante vom Ausfluss reflektierte Verteilungen. Das führt zu einer vereinfachten Ausfluss-Randbedingung:

$$f_i = w_i c_A \tag{2.22}$$

Diese benötigt weniger Rechenzeit, da die Verteilungen nur einmalig zu Beginn der Simulation berechnet werden. Trifft die Annahme jedoch nicht zu, kann das zu einer Verletzung der Massenerhaltung führen.

3. Kopplungsmethoden und Implementierung

Koppelt man eine LBM-Simulation mit einer Festkörper-Engine kommen verschiedene Probleme auf, für die man Lösungen finden muss. Zuerst müssen die Festkörper in das LBM-Gitter eingebettet werden, damit diese von der LBM-Simulation berücksichtigt werden können. Dabei wird der Festkörper als Wand gesehen. Da dieser sich nicht genau dem Gitter anpasst, benötigt man spezielle Randbedingungen. Eine Folge der Bewegung der Festkörper ist, dass Fluidzellen reaktiviert werden, welche passend initialisiert werden müssen. Das schließt den Einfluss des Festkörpers auf das Fluid ab. Man benötigt noch den Einfluss des Fluids auf den Festkörper. Dazu ist eine Berechnung der Kraft aus der LBM-Simulation notwendig. Diese muss noch in Translation und Rotation umgerechnet werden. In Abbildung 3.1 werden die Vorgänge dargestellt.

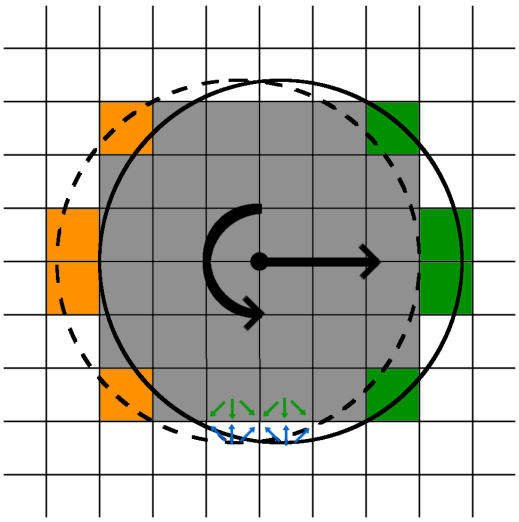


Abbildung 3.1.: Veranschaulichung der Kopplungsvorgänge. Graue Gitterzellen sind inaktiv, orange müssen initialisiert und grüne deaktiviert werden. Blaue Pfeile sind Verteilungen, die auf den Festkörper treffen, grüne die reflektierten Verteilungen. Schwarze Pfeile zeigen die Bewegungen des Festkörpers

In dieser Arbeit wird sich auf den 2D-Fall und auf kreisförmige Festkörper beschränkt.

3.1. Einbinden der Festkörper in die LBM-Simulation

Das Einbinden der Festkörper erfolgt auf zwei Ebenen. Einerseits muss die Festkörper-Engine in den Simulationsablauf eingebunden werden, andererseits müssen die Festkörper bei den LBM-Berechnungen berücksichtigt werden. Die Festkörper-Engine ist hierbei für das Aktualisieren der Positionen und Geschwindigkeiten der Festkörper zuständig.

Die Lattice Boltzmann Methode arbeitet mit Gitterzellen. Gitterzellen, die nun aber von einem Festkörper bedeckt sind, müssen für die Berechnungen ignoriert werden, da die LBM-Simulation nur mit Fluidzellen arbeiten kann. Das liegt daran, dass die Lattice Boltzmann Methode die Navier-Stokes Gleichung approximiert und damit nur für Fluidzellen definiert ist. Daher muss man Zellen für die LBM-Berechnungen je nach Position der Festkörper deaktivieren bzw. reaktivieren. In diesem Zusammenhang werden Zellen im Folgenden auch als aktiv/inaktiv klassifiziert. Da die Ränder von Festkörpern selten perfekt auf den Zellgrenzen liegen, erfolgt die Auswahl der inaktiven Zellen mithilfe eines Algorithmus. Dabei muss darauf geachtet werden, dass dieser konform mit den Kopplungsmethoden arbeitet. Dies ist insbesondere für die Randbedingungen wichtig, da diese teilweise davon ausgehen, dass sich der tatsächliche Rand zwischen einem inaktiven und aktiven Gitterpunkt befindet. Hier wurden alle Zellen, deren Zellmittelpunkt innerhalb des Festkörpers liegt, deaktiviert (siehe Abbildung 3.2). Der Zellmittelpunkt der Gitterzelle entspricht dem Gitterpunkt des LBM-Gitters. Über diese Definition ist die Zuweisung der Gitterzellen einfach zu erreichen.

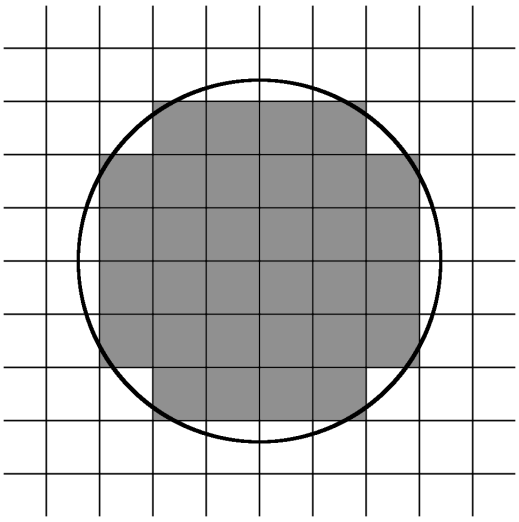


Abbildung 3.2.: Aktive/Inaktive Gitterzellen bei einem kreisförmigen Festkörper auf dem Gitter. Inaktive Gitterzellen sind grau.

Für die LBM-Berechnungen müssen bestimmte Eigenschaften der Festkörper vorhanden sein. Diese müssen vor den Berechnungen übertragen werden und sind:

- Position
- Radius

- Geschwindigkeit

Position und Radius werden benötigt um die Gitterpunkte dem Festkörper zuzuweisen. Die Geschwindigkeit verwendet man für die Initialisierung reaktivierter Fluidpunkte hinter dem Festkörper. Zusätzlich müssen nach den Berechnungen Daten an die Festkörper-Engine übergeben werden:

- Kraft auf Festkörper
- Drehmoment der Festkörper

Kraft und Drehmoment werden während der LBM-Simulation berechnet, da dort alle notwendigen Daten vorhanden sind. Die Festkörper-Engine verwendet diese, um den translatorischen und den rotatorischen Anteil der Bewegung zu bestimmen.

Die explizite Aktualisierung der Festkörper in der Festkörper-Engine ermöglicht, die Festkörper-Eigenschaften nur alle n LBM-Zeitschritte zu bestimmen. Die Größe von n ist abhängig von der maximalen Geschwindigkeit der Festkörper. Zu jeder Zeit muss gewährleistet sein, dass die Festkörper sich nicht um mehr als eine Zellgröße verschieben, da nur eine Reihe von Zellen in jedem Schritt initialisiert werden kann. Genaueres hierzu in Abschnitt 3.3.4.

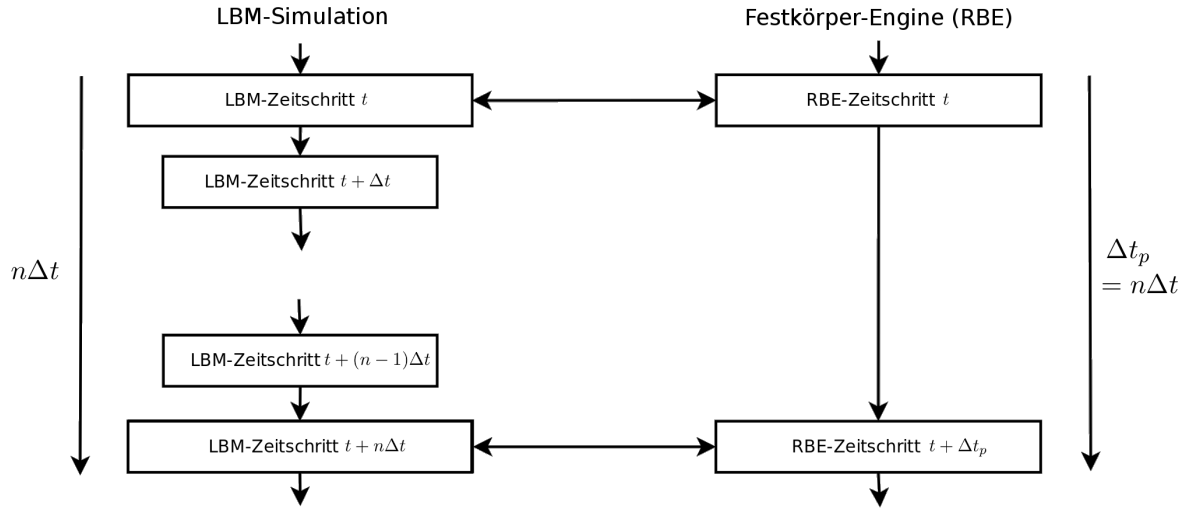


Abbildung 3.3.: Kopplung des Programmablaufs der LBM-Simulation und der Festkörper-Engine.

Abbildung 3.3 zeigt einen Ausschnitt des seriellen Programmablaufs, der die Einbindung der Festkörper-Engine und dessen Interaktion mit der LBM-Simulation zeigt.

3.2. Randbedingungen

Bewegte Festkörper benötigen bewegliche Ränder. Die entsprechenden Randbedingungen wurden bereits in Abschnitt 2.2.2 vorgestellt. Das on-grid Verfahren kann dabei direkt auf die deaktivierten Gitterpunkte angewendet werden.

Genauere Ergebnisse liefert eine Interpolation über umliegende Fluidgitterpunkte mit Berücksichtigung des Abstands zum Festkörper. Dadurch wird die tatsächliche Form des Festkörpers betrachtet, anstatt die auf die Gitterzellen zugeschnittene. Dazu wird der Abstand q vom Fluidgitterpunkt zum Rand des Festkörpers in Richtung der Geschwindigkeit $\mathbf{c}_i$ benötigt. q liegt auf dem Intervall $(0, 1]$, da für $q \leq 0$ der Fluidgitterpunkt inaktiv ist und für $q > 1$ ein anderer Fluidpunkt zwischen dem aktellen und dem Festkörper-Rand liegt. Ein einfaches, geometrischen Verfahren zur Berechnung von q kann man [ITR08] entnehmen.

Mit diesem q können die Verteilungen berechnet werden, die der Rand reflektiert. Hierfür wird die in [BFL01] vorgestellte Methode um den Einfluss der Dichte auf den Geschwindigkeitsterm erweitert:

$$
\begin{aligned}
f_{\bar{i}}(\mathbf{x}, t+dt) &= 2q f_i^*(\mathbf{x}, t) + (1-2q) f_i^*(\mathbf{x}-\mathbf{c}_i, t) - 2w_i \rho_w \tfrac{1}{c_s^2} \mathbf{c}_i \mathbf{v}_w && q < \tfrac{1}{2} \\
f_{\bar{i}}(\mathbf{x}, t+dt) &= \tfrac{1}{2q} f_i^*(\mathbf{x}, t) + \tfrac{2q-1}{2q} f_{\bar{i}}^*(\mathbf{x}, t) - \tfrac{1}{q} w_i \rho_w \tfrac{1}{c_s^2} \mathbf{c}_i \mathbf{v}_w && q \geq \tfrac{1}{2}
\end{aligned}
\tag{3.1}
$$

Dabei werden zwei Fälle unterschieden: $q < \frac{1}{2}$ und $q \geq \frac{1}{2}$. Das liegt daran, dass die zum Fluidgitterpunkt reflektierten Partikel aufgrund des diskreten $\Delta\mathbf{x}$ ihren Ursprung zwischen zwei Gitterpunkten haben müssten. Da Populationen allerdings nur auf diesen definiert sind, werden die Verteilungen interpoliert. Je nachdem zwischen welchen zwei Gitterpunkten der Urprung der Verteilungen sein müsste, wird dann eine andere Formel benötigt. Diese Problematik wird in Abbildung 3.4 veranschaulicht.

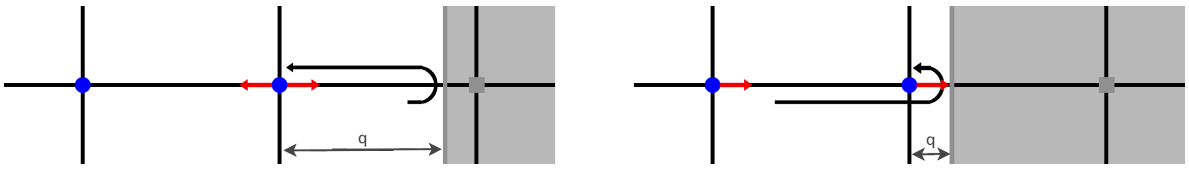


Abbildung 3.4.: Darstellung der Interpolationsproblematik bei den Randbedingungen für $q > 0.5$ (links) und $q < 0.5$ (rechts). Aktive Gitterpunkte sind blau, inaktive grau. Die schwarzen Pfeile zeigen den Weg der fiktiven zum Fluidgitterpunkt reflektierten Partikel. Rote Pfeile symbolisieren die Verteilungen die Interpoliert werden (siehe auch Abbildung 2.2).

Eine weitere Methode wird in [ITR08] vorgestellt. Diese vereint die beiden Interpolationsformeln und kommt dadurch mit nur einer Formel für die Randabstoßung aus. Die Gleichung wird dafür an die hier verwendeten LBM-Formeln angepasst:

$$
\begin{aligned}
f_{\bar{i}}(\mathbf{x}, t+dt) = \tfrac{1}{1+q} \quad & [(1-q) \cdot f_i^*(\mathbf{x}-\mathbf{c}_i, t) \\
& +q \cdot f_i^*(\mathbf{x}, t) \\
& +q \cdot f_{\bar{i}}^*(\mathbf{x}, t) \\
& -2w_i \rho_w \tfrac{1}{c_s^2} \mathbf{c}_i \mathbf{v}_w \big]
\end{aligned}
\tag{3.2}
$$

3.3. Initialisieren von neuen Fluidzellen

Da sich die Festkörper durch das Fluid bewegen, ändern auch die Zellen ihren Zustand von aktiv nach inaktiv und wieder zu aktiv zurück. Dies wird in Abbildung 3.5 veranschaulicht.

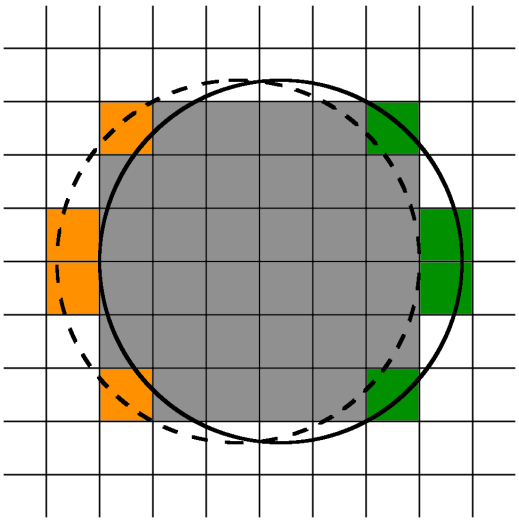


Abbildung 3.5.: Wechsel der Gitterzellen von aktiv nach inaktiv (grün) und von inaktiv nach aktiv (orange).

Wechselt eine Zelle ihren Zustand von Fluid zu Festkörper, kann man ihren Inhalt einfach löschen. Das liegt daran, dass das enthaltene Fluid durch die Randbedingungen (Abschnitt 3.2) schon verdrängt wurde und inaktive Zellen keine Werte benötigen.

Wechselt eine Zelle ihren Zustand von Festkörper zu Fluid, enthält sie keine sinnvollen Werte für die LBM-Simulation. Damit diese trotzdem stabil läuft, müssen die Zellen mit passenden Werten initialisiert werden. Dafür gibt es verschiedene Methoden, die sich sowohl in ihrer Genauigkeit, als auch in ihrer Komplexität unterscheiden.

3.3.1. Standardinitialisierung

Die Standardinitialisierung belegt die Zelle mit Default-Werten. Es wird kein Bezug auf den umgebenden Fluss genommen, was diese Methode sehr schnell, aber auch sehr ungenau macht. Die Verteilungen werden wie folgt gesetzt:

$$f_i(\mathbf{x}, t) = w_i \tag{3.3}$$

Das sind die Verteilungen, die für den Kollisionsschritt (Gleichung (2.2)) im Zeitschritt t verwendet werden. Dadurch ergeben sich folgende Werte für die Parameter der Equilibriumsverteilung f^{eq}:

$$\rho = \sum f_i = \sum w_i = 1 \tag{3.4}$$

$$\mathbf{u} = \frac{1}{\rho} \sum f_i \mathbf{c}_i = (0,0)^T \tag{3.5}$$

Indem man die Dichte auf 1 und die Geschwindigkeit auf 0 setzt, erhält man die einfachste, stabile Initialisierung einer Zelle. Diese Methode nimmt an, dass der lokal eingeführte Fehler ausreichend klein ist, so dass er sich schnell dem lokalen Gleichgewicht anpasst.

3.3.2. Equilibriumsinitialisierung

Eine Methode, die den Fluss und den Festkörper berücksichtigt, ist die Equilibriumsinitialisierung. Diese wird in [Neu13] erläutert. Dabei wird die Initialisierung in folgende Schritte aufgeteilt:

1. Interpolieren der Dichte

   Für die Dichte wird das arithmetische Mittel der Dichten der umliegenden Zellen verwendet. Dabei gilt es zu beachten, dass nicht in jeder umliegenden Zelle auch eine Dichte vorhanden ist, da diese teilweise vom Festkörper verdeckt und daher inaktiv sind.

2. Bestimmen der Geschwindigkeit des Flusses über die Geschwindigkeit des Festkörpers

   Für die Geschwindigkeit übernimmt man die des Festkörpers am nächsten Oberflächenpunkt. Da hier nur kreisförmige Festkörper verwendet werden, entspricht die Geschwindigkeit an einem beliebigen Oberflächenpunkt der des Festkörpers.

3. Berechnung der einkommenden Verteilungen

   Mit Dichte und Geschwindigkeit sind alle Parameter für die Equilibriumsverteilung vorhanden. Diese wird für die Initialisierung der Verteilungen verwendet.

   $$f_i = f_i^{eq}(\mathbf{u}_{closest}, \rho_{int}) \tag{3.6}$$

Die erhaltenen Verteilungen f_i werden im Kollisionsschritt verwendet.

3.3.3. Non-Equilibriumsinitialisierung

Die Non-Equilibriumsinitialisierung funktioniert im Prinzip wie die Equilibriumsinitialisierung. Der Unterschied besteht darin, dass im letzten Schritt bei der Berechnung der Verteilungen auch ein Non-Equilibriumsteil miteinbezogen wird.

Bei einem normalen LBM-Schritt werden die einkommenden Verteilungen in einem Kollisionsschritt aus Non-Equilibriums- und Equilibriumsverteilungen berechnet. Bei der Equilibriumsinitialisierung sind diese einkommenden Verteilungen nicht vorhanden. Daher werden sie mit den speziell berechneten Equilibriumsteil approximiert. In dieser Approximation fehlt allerdings der Non-Equilibriumsteil

komplett und ist daher zwangsläufig nicht optimal. Bei der Non-Equilibriumsinitialisierung wird daher für die Schätzung der einkommenden Verteilungen zusätzlich noch ein Non-Equilibriumsteil verwendet. Diesen erhält man durch Extrapolation. Hier wird eine Extrapolation ersten Grades verwendet, da auch höhere Grade kein besseres Ergebnis liefern würden (siehe [Cai08]). Extrapolieren ersten Grades entspricht dem Kopieren der Non-Equilibriumsverteilungen eines benachbarten Gitterpunktes. Die einkommenden Verteilungen werden dann mit dem arithmetischen Mittel von diesen und den Equilibriumsverteilungen bestimmt:

$$f_i = \frac{f_i^{eq}(\mathbf{u}_{closest}, \rho_{int}) + f_i^{neqex}}{2} \tag{3.7}$$

f_i^{neqex} sind die durch Extrapolation erhaltenen Non-Equilibriumsverteilungen. Der hier verwendete Mittelwert entspricht einem Kollisionsschritt (Gleichung 2.2) mit $\tau = 2$ und ist damit konform mit dem Gedanken, dass die einkommenden Verteilungen aus einem Kollisionsschritt des letzten Zeitschrittes stammen.

3.3.4. Grenzen der Initialisierung

Wie bereits in Abschnitt 3.1 angedeutet kann man auf diese Weise nicht beliebig viele Zellen initialisieren. Das hängt mit der Interpolation der Dichten zusammen. Diese geht davon aus, dass in manchen der umliegenden Zellen sinnvolle Werte vorhanden sind. Sobald sich der Festkörper jedoch so schnell bewegt, dass mehr als eine Reihe von zu initialisierenden Zellen entsteht, kann dies nicht mehr gewährleistet werden (siehe Abbildung 3.6).

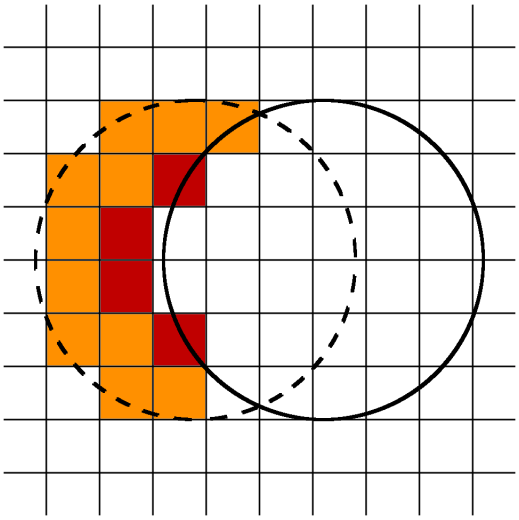


Abbildung 3.6.: Initialisierung von Zellen in der zweiten Reihe. Orange: Zellen die normal initialisiert werden können, Rot: kritische Zellen die keine Daten von umliegenden interpolieren können.

Bei der Standardinitialisierung gibt es dieses Problem auch, da diese davon ausgeht, dass sich die Werte einpendeln. Dies wird allerdings stark beeinträchtigt, wenn Zellen in mehreren Reihen initialisiert werden.

3.4. Krafteinwirkung auf die Festkörper

Um die Kraft zu bestimmen, die das Fluid auf den Festkörper ausübt, wird die in [ITR08] vorgeschlagene Methode verwendet. Diese benutzt nur bereits vorhandene Komponenten und ist damit einfach und schnell zu berechnen.

$$\mathbf{F}_P = \sum_{\mathbf{x}_P} \sum_{i=0}^{8} \mathbf{c}_i [f_i(\mathbf{x}_P, t) + f_{\bar{i}}(\mathbf{x}_f, t)] \Delta\mathbf{x}/\Delta t \tag{3.8}$$

$f_{\bar{i}}(\mathbf{x}_f, t)$ sind die Fluidverteilungen, die auf den Festkörper P treffen, wobei $\mathbf{x}_f = \mathbf{x}_P - \mathbf{c}_i \Delta t$. $f_i(\mathbf{x}_P, t)$ sind die dazugehörigen reflektierten Verteilungen. Diese kann man in Abbildung 3.7 sehen. Ist $\mathbf{x}_f$

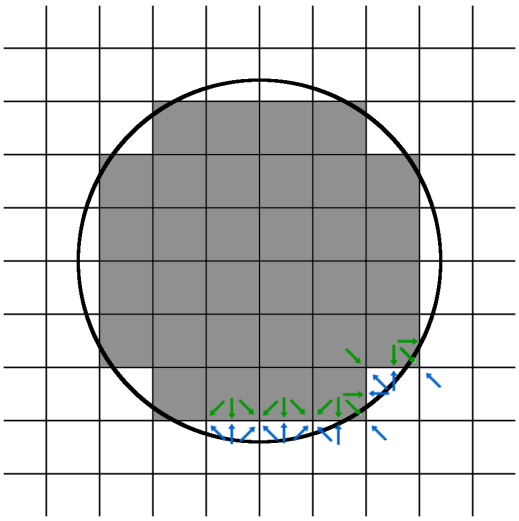


Abbildung 3.7.: Die auf den Festkörper treffenden (blau) und die zugehörigen reflektierten Verteilungen (grün) beispielhaft für einige Gitterzellen.

kein Fluidgitterpunkt, sind beide Verteilungen 0.

3.5. Bewegen der Festkörper

Grundsätzlich gibt es bei der Bewegung eines Festkörpers in einem Fluid zwei Arten zu unterscheiden: die Translation und die Rotation um die eigene Achse.

3.5.1. Translation

Um diese Bewegung zu berechnen, muss zunächst die Beschleunigung mit Hilfe des zweiten Newton'schen Gesetzes ermittelt werden.

$$\mathbf{a}_{P,t} = \frac{\mathbf{F}_{P,t}}{m_P} \tag{3.9}$$

$\mathbf{F}_{P,t}$ ist die Kraft $\mathbf{F}_P$ aus Gleichung (3.8) im Zeitschritt t, m_P die Masse des Festkörpers P. Damit kann nun auch die Geschwindigkeit und die neue Position des Festkörpers berechnet werden.

$$\mathbf{v}_{P,t+\Delta t_P} = \mathbf{v}_{P,t} + \mathbf{a}_{P,t} \cdot \Delta t_p \tag{3.10}$$

$$\mathbf{x}_{P,t+\Delta t_P} = \mathbf{x}_{P,t} + \mathbf{v}_{P,t+\Delta t_P} \cdot \Delta t_p \tag{3.11}$$

Dabei ist Δt_p der Zeitschritt der Festkörper. Für die Berechnungen benötigt man eine geeignete Initialisierung der Position $\mathbf{x}_{P,0}$ und Geschwindigkeit $\mathbf{v}_{P,0}$. Die initiale Position muss so gewählt werden, dass der Festkörper sich innerhalb der Fluid-Domäne befindet, und die initiale Geschwindigkeit darf nicht zu groß sein, da die Simulation sonst instabil wird.

3.5.2. Rotation

Um die Rotation eines Festkörpers um die eigene Achse zu bestimmen, benötigt man das Drehmoment. Dieses wird aus der Kraft $\mathbf{F}_{\mathbf{x}_P}$ auf den Festkörpergitterpunkt $\mathbf{x}_P$ und dem Vektor $\mathbf{d}_{so_{\mathbf{x}},P}$ vom Schwerpunkt des Festkörpers zum Randpunkt, an dem die Kraft anliegt, berechnet. $\mathbf{F}_{\mathbf{x}_P}$ leitet sich von der Gleichung der Krafteinwirkung (3.8) her und wird zum Zeitpunkt t wie folgt bestimmt:

$$\mathbf{F}_{\mathbf{x}_P,t} = \sum_{i=0}^{8} \mathbf{c}_i [f_i(\mathbf{x}_P, t) + f_{\bar{i}}(\mathbf{x}_f, t)] \tag{3.12}$$

Der Distanzvektor wird aus dem Schwerpunkt $\mathbf{s}_P$ und den Koordinaten des Festkörpergitterpunktes $\mathbf{x}_P$ berechnet und muss noch auf die passende Größe skaliert werden, da die Koordinaten der Gitterpunkte diskret sind. Bei kreisförmigen Festkörpern sind alle Oberflächenpunkte gleich weit entfernt. Als Skalierungsfaktor wird daher der Radius des Festkörpers r_P verwendet.

$$\mathbf{d}_{so_{\mathbf{x}},P} = \frac{\mathbf{x}_P - \mathbf{s}_P}{|\mathbf{x}_P - \mathbf{s}_P|} \cdot r_P \tag{3.13}$$

Damit wird der Drehmoment mit folgender Gleichung berechnet, die eine auf den 2D-Fall vereinfachte Variante der aus [Neu13] ist:

$$\tau_{P,t} = \sum_{\mathbf{x}_P} ((\mathbf{d}_{so_{\mathbf{x}},P})_x (\mathbf{F}_{\mathbf{x}_P,t})_y - (\mathbf{d}_{so_{\mathbf{x}},P})_y (\mathbf{F}_{\mathbf{x}_P,t})_x) \tag{3.14}$$

$(\cdot)_x$ liefert dabei die x-Komponente eines Vektors. $\sum_{\mathbf{x}_P}$ iteriert über die Festkörpergitterpunkte. Es muss noch das Trägheitsmoment des Festkörpers I_P bestimmt werden. Für eine Scheibe gilt:

$$I_P = \frac{m_P r_P^2}{2} \tag{3.15}$$

Nun kann die Winkelbeschleunigung $\alpha_{P,t}$ berechnet werden.

$$\alpha_{P,t} = \frac{\tau_{P,t}}{I_P} \tag{3.16}$$

Damit kann Winkelgeschwindigkeit ω_P und Winkel ϕ_P für den nächsten Zeitschritt bestimmt werden.

$$\omega_{P,t+\Delta t_P} = \omega_{P,t} + \alpha_{P,t} \cdot \Delta t_P \tag{3.17}$$

$$\phi_{P,t+\Delta t_P} = \phi_{P,t} + \omega_{P,t+\Delta t_P} \cdot \Delta t_P \tag{3.18}$$

Der Winkel $\phi_{P,t+1}$ ist die Ausrichtung des Festkörpers P zum Zeitschritt $t+1$ und liegt im Intervall $[0, 360)$. Erhält man einen Wert außerhalb dieses Intervalls, wird dieser mithilfe des Modulo-Operators korrigiert.

3.6. Implementierungshinweise

Festkörper im LBM-Gitter Die LBM-Simulation erfolgt auf einem $n \times m$-Zellgitter. Jede dieser Zellen hat einen Typ (Fluid, Rand, Einfluss, Ausfluss). Der Typ muss zur Abfrage gespeichert sein. Dafür wird ein Flag-Gitter verwendet, in welchem unter der Position jeder Zelle ihr Typ kodiert ist. Beim Einbinden der Festkörper muss auch dieses Flag-Gitter erweitert werden. Dafür bieten sich zwei Möglichkeiten an:

1. Integration in das vorhandene Flag-Gitter mit einer neuen Flag
2. Neues Flag-Gitter, das zwischen Festkörper und nicht Festkörper unterscheidet

Das Problem bei der Integration ist, dass sich die Festkörper ständig bewegen. Dadurch müssen die Festkörper-Flags nach jedem Festkörper-Zeitschritt aktualisiert werden. Die Standardflags der LBM-Simulation ändern sich nur, wenn das Gebiet manuell z.B. durch das Einfügen von Rändern verändert wird.

Für den Vergleich der beiden Methoden muss die Performance von Update und Typabfrage in Betracht gezogen werden.

Bei Methode 1 müssen bei einem Update die alten Festkörperflags entfernt und die neuen gesetzt werden. Unter der Annahme, dass die Iteration über die Festkörperflags zum Entfernen bzw. Setzen die Laufzeit $\mathcal{O}(k)$ hat, ist die Laufzeit des Update-Vorgangs $2 \cdot \mathcal{O}(k)$. Dabei ist k die Anzahl der Festkörpergitterpunkte. Methode 2 schafft das Update dagegen in $\mathcal{O}(k)$, da die Flags nicht auf den Ursprungswert zurückgesetzt werden müssen. Dies erreicht man u.a. mit aufsteigenden Flagwerten. Dafür wird bei jedem Update der aktuelle Flagwert erhöht und die Festkörperflags damit gesetzt. Dann gilt, dass alle Flags ungleich dem aktuellen Flagwert als nicht Festkörper gelten. Das Löschen der Flags ist erst notwendig, wenn der aktuelle Flagwert das Maximum erreicht hat und wieder auf 0 springt.

Die Typabfrage bei Standard-LBM erfolgt mithilfe von Switch-Case. Methode 1 fügt in diese einen zusätzlichen Case ein. Wird dieser am Ende eingefügt, hat man Mehrkosten bezüglich der ungekoppelten LBM-Simulation in $\mathcal{O}(k)$. Bei Methode 2 benötigt man zusätzlich eine if-Abfrage. Setzt man diese um die Switch-Case Anweisung, hat man Mehrkosten in $\mathcal{O}(n \cdot m)$ (da $n \times m$-Gitter).

Da die relativen Kosten (relativ zu k bzw n, m) des Updates höher sind als die der Typabfrage, gilt: Solange k nicht sehr klein ist, hat Methode 2 eine bessere Performance als Methode 1.

Parallelität Aufgrund der Gitterstruktur ist die LBM-Simulation perfekt für Parallelisierung geeignet. Das Einbinden der Festkörper führt jedoch dazu, dass für manche Berechnungen, z.B. die der Interpolation der Dichte bei der Initialisierung (siehe Abschnitt 3.3), Daten aus anderen Zellen benötigt werden. Dieses Problem wird hier gelöst, indem alle nötigen Daten aus dem vorherigen Zeitschritt verwendet werden, anstatt an den Gitterpunkten, an denen die Daten schon berechnet wurden, die Werte des aktuellen Zeitschrittes zu nutzen. Dies senkt zwar die Genauigkeit der Simulation, macht diese aber auch einfacher und damit schneller durchführbar. Das ist insbesondere wichtig für die Echtzeitfähigkeit.

Kraftberechnung Die Gleichung zur Berechnung der Kraft auf den Festkörper (3.8) ist eine Summe über alle Festkörpergitterpunkte. Da diese in der LBM-Simulation ignoriert werden, müsste man am Ende eines Zeitschrittes noch extra die Kraft bestimmen. Um den Zeitaufwand der LBM-Simulation so gering wie möglich zu halten, wird die Kraftberechnung integriert. Betrachtet man die Gleichung genauer, sieht man, dass alle verwendeten Variablen auch bei der Berechnung der reflektierten Verteilungen vorkommen (Abschnitt 3.2). Dadurch können die jeweiligen Komponenten der Kraft direkt mit der Randabstoßung am Festkörper bestimmt werden.

$$\mathbf{F}_P = \sum_{f_{\bar{i}}(\mathbf{x}_f,t)} \mathbf{c}_i[f_i(\mathbf{x}_P,t) + f_{\bar{i}}(\mathbf{x}_f,t)]\Delta\mathbf{x}/\Delta t \tag{3.19}$$

Bindet man diese Gleichung korrekt in die Simulation - nämlich nach dem Kollisionsschritt - ein, erhält man:

$$\mathbf{F}_P = \sum_{f^*_{\bar{i}}(\mathbf{x}_P,t-\Delta t)} \mathbf{c}_i[f^*_i(\mathbf{x}_f,t-\Delta t) + f^*_{\bar{i}}(\mathbf{x}_P,t-\Delta t)]\Delta\mathbf{x}/\Delta t \tag{3.20}$$

Auf diese Art und Weise wird auch die Berechnung des Drehmoments in den Simulationsablauf integriert.

4. Anwendung und Analyse

4.1. Strömungssimulation Talfer

Talfer ist ein 2D-Spiel, bei dem der Spieler versucht Festkörper in einen Zielbereich zu bewegen. Allerdings existiert auf dem Spielfeld eine Strömung, die die Festkörper von diesem Zielbereich ablenken bzw. an ihm vorbeiführen. Mithilfe einer Kinect-Steuerung kann der Spieler Einfluss auf die Festkörper nehmen, indem er durch Armbewegungen Kräfte auf sie erzeugt. Die Strömung wird mit der Lattice Boltzmann Methode simuliert, wie in Kapitel 2 erläutert. Im Orginal-Talfer ist die LBM-Simulation nur einseitig mit der Festkörper-Engine gekoppelt. Diese berechnet nämlich die Krafteinwirkung auf die Festkörper über die Geschwindigkeit des darunter liegenden Fluids, also der Fluidgeschwindigkeit $\mathbf{u}_f$ an dem Gitterpunkt der am nächsten zum Mittelpunkt des Festkörpers liegt.

$$\mathbf{F_P} = \frac{\mathbf{u}_f}{|\mathbf{u}_f|} \cdot \frac{\rho_f}{2} |\mathbf{u}_f|^2 A_P \tag{4.1}$$

A_P ist der Flächeninhalt des Festkörpers P. Die Rotation wird auch mithilfe des darunterliegenden Fluids bestimmt. Der Festkörper hat jedoch keinen Einfluss auf das Fluid. Diese Vorgänge werden nun von den in Kapitel 3 beschriebenen Kopplungsverfahren ersetzt.

Das Spielfeld Das Spielfeld wird zu einem Gitter diskretisiert. Dabei hat jeder Gitterpunkt einen Typ. Die möglichen Typen sind: Wand, Fluid, Einfluss, Ausfluss, Ziel. Die Größe und Typkonfiguration des Spielfeld-Gitters wird aus einer Bilddatei ausgelesen, indem bestimmten Farben ein bestimmter Typ zugewiesen wird. Das Spielfeld-Gitter wird dann als Flag-Field für die LBM-Simulation verwendet. Ein Beispiel-Spielfeldbild ist in Abbildung 4.1 zu sehen.

Visualisierung Da Talfer auf Fluidströmung in 2D basiert, wird eine Visualisierung benötigt, die diese Strömungen zeigt. Dafür wird die Richtung der Fluidgeschwindigkeiten mithilfe von Pfeilen dargestellt und deren Größe farblich kodiert. Blau steht dabei für eine niedrige Geschwindigkeit, Rot für eine hohe. Abbildung 4.2 zeigt einen Frame der Ausgabe von Talfer.

Als Input wird das Spielfeld als Flag-Field, die Fluidgeschwindigkeiten aus der LBM-Simulation und die Festörper-Daten aus der Festkörper-Engine benötigt. Der erzeugte Frame wird dann als Ausgabe an den Video-Output übergeben.

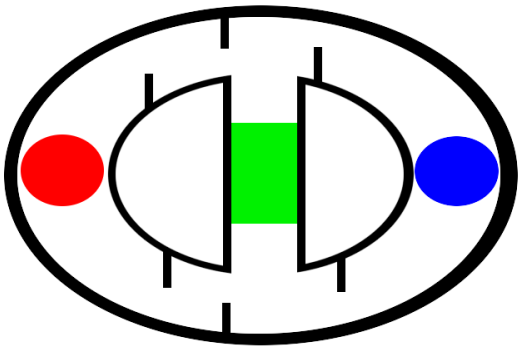


Abbildung 4.1.: Beipiel eines Bildes, das in ein Spielfeld konvertiert werden kann. (Schwarz: Wand, Weiß: Fluid, Rot: Einfluss, Blau: Ausfluss, Grün: Ziel).

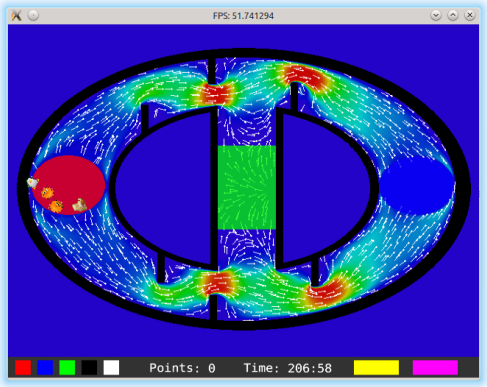


Abbildung 4.2.: Standbild eines laufenden Talfer-Spiels.

Die vollständige Pipeline Die einzelnen Teile des Programms werden in einer Pipeline zusammengefügt. Diese Pipeline besteht aus folgenden Modulen:

- Image Input

  liest die Bilddatei des Spielfelds ein. Die Bilddatei muss dabei der besprochenen Farbcodierung entsprechen.

- Image Processing

  wandelt die Bilddaten in ein verwendbares Flag-Field um. Dabei werden die einzelnen Farben in die dazugehörigen Flags konvertiert. Jeder Pixel wird ein Punkt des Flag-Fields.

- LBM

führt die Simulation der Lattice Boltzmann Methode durch. Dabei werden während eines Simulationsdurchlaufes mehrere LBM-Zeitschritte ausgeführt. Das Flag-Field wird in ein LBM-internes Gitter konvertiert.

- RBE

  implementiert die Festkörper-Engine. Sie berechnet Winkelgeschwindigkeit und Kraft für jeden Festkörper. Die Festkörper bestimmen dann mithilfe der seit der letzten Aktualisierung vergangenen Zeit Position und Ausrichtung.

- Kinect

  erlaubt dem Spieler eine Interaktion mit dem Programm. Er kann mit Handbewegungen Kräfte auf die Festkörper ausüben.

- Visualisierung

  wandelt die berechneten Daten in ein Bild um. Dabei werden u.a. die Richtungspfeile der Strömung erzeugt und die Kinect-Kräfte dargestellt. Das Flag-Field wird hier wieder in Bilddaten konvertiert.

- VideoOutput

  gibt das erzeugte Bild auf dem Bildschirm aus. Dafür wird OpenGL verwendet.

Abbildung 4.3 zeigt die Pipeline.

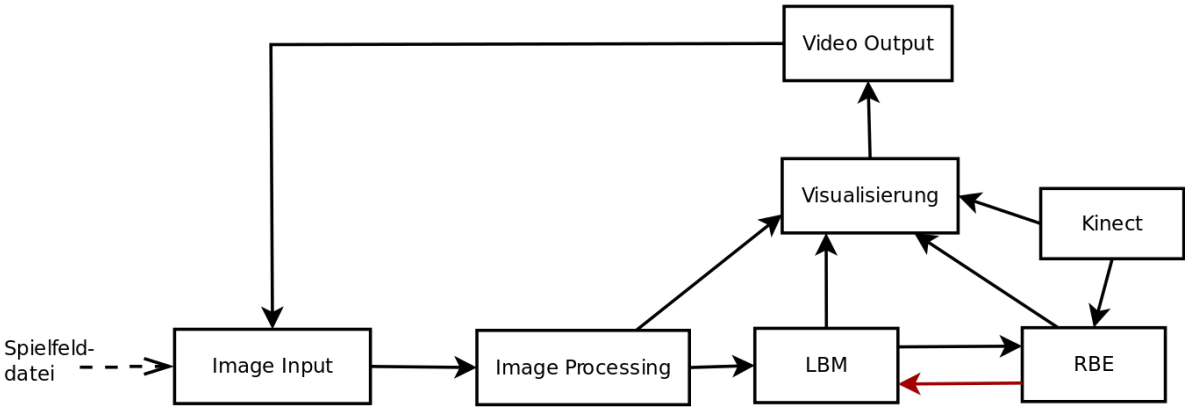


Abbildung 4.3.: Pipeline der Strömungssimulation Talfer. Beginn ist bei „Image Input“. Bereits vorhandene Komponenten sind schwarz, für die Kopplung eingefügte rot.

Damit die Kopplungsmethoden korrekt implementiert werden konnten, wurde die Pipeline des Original Talfers erweitert.

4.2. Analyse der Kopplungsmethoden

Die Analyse der Kopplungsmethoden gliedert sich wie folgt:

1. Verifikation
2. Vergleich der Verfahren
3. Laufzeitanalyse

Die Verifikation überprüft, ob unser Verfahren angemessene Ergebnisse liefert. Bevor man darauf eingehen kann, welches der Verfahren sich besser eignet, muss gezeigt werden, dass sie sich überhaupt wie erwartet verhalten.

Im Vergleich der Verfahren wird überprüft, welche der unterschiedlichen Verfahren, die in Abschnitt 3 vorgestellt wurden, sich besser eignet.

Bei der Laufzeitanalyse werden die Auswirkungen der Kopplungsverfahren auf die Laufzeit untersucht.

Davor müssen aber noch die Modalitäten geklärt werden.

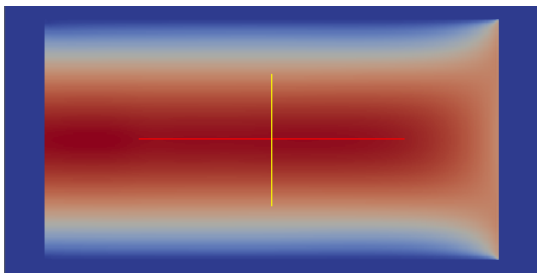
4.2.1. Testmethodik

Für die Tests wurden die jeweiligen Kopplungsmethoden im bereits vorgestellten Talfer umgesetzt. Als Feld wurde ein Kanal gewählt. Dieser besteht in x-Richtung aus 2,5% feste Wand, 5% Einfluss, 85% Fluid, 5% Ausfluss und dann wieder 2,5% feste Wand. In y-Richtung ist 5% feste Wand, 90% Fluid und wieder 5% feste Wand. Das Verhältnis von Breite zu Höhe ist 2:1. Abbildung 4.4 zeigt den hier spezifizierten Kanal.

Abbildung 4.4.: Das Feld, das im folgenden als Testkanal für die Analyse der Methoden im Talfer verwendet wird. Der Farbcode ist analog zu dem in Abschnitt 4.1.

In diesem Kanal wird dann ein Poiseuille-Fluss erzeugt. Für den Ausfluss werden die vereinfachten Randbedingungen benutzt. Die dafür verwendeten Standardparameter sieht man in Tabelle 4.1. Abbildung 4.5 veranschaulicht die entstehende Strömung.

In diesen bestehenden Poiseuille-Fluss wurde dann ein Festkörper eingefügt. Wenn nicht extra angegeben, wurde als Standardverfahren für die Randbedingungen die Methode von [ITR08] und für



Visualisierung des Poiseuille-Flusses im Kanal.

Abbildung 4.5.: Visualisierung des Poiseuille-Flusses im Kanal. Die Farbskala zeigt die Größe der Flussgeschwindigkeit. Blau steht für niedrige Geschwindigkeit, rot für hohe.

Konstante	Beschreibung	Wert
U_m	max. Geschwindigkeit für den Poiseuille-Fluss	0.3
ρ_E	Dichte am Einfluss	1.4
c_A	Ausflusskonstante	0.8
τ	Relaxationszeit	0.66
W	Breite des Feldes	400
H	Höhe des Feldes	200
$\mathbf{x}_0$	Startposition des Festkörpers	$(60.0, 100.0)^T$
$\mathbf{v}_0$	Startgeschwindigkeit des Festkörpers	$(0.0, 0.0)^T$
r_P	Radius des Festkörpers	10

Tabelle 4.1.: Standardparameter für den bei den Tests verwendeten Poiseuille-Fluss.

die Initialisierung die Equilibriumsinitialisierung verwendet. Die Daten des Rechners, auf dem die Tests durchgeführt wurden, zeigt Tabelle 4.2.

4.2.2. Verifikation

Bevor die Verfahren individuell verglichen werden können, muss die Korrektheit von diesen verifiziert werden. Dabei werden zwei Teile unterschieden: Die Verifikation der LBM-Fluidsimulation und die der Festkörper.

Art	Verwendet
Prozessor	Intel Quad Core i3-3225 CPU @ 3.30GHz
Arbeitsspeicher	4Gb

Tabelle 4.2.: Relevante Daten des Rechners, der für die Tests verwendet wurde.

Für die Verifikation der LBM-Fluidsimulation werden folgende Punkte untersucht:

- Massenerhaltung
- Impulserhaltung
- Strömungsbild

Die Notwendigkeit der Massen- und Impulserhaltung leitet sich von den Navier-Stokes-Gleichungen ab. Das Strömungsbild soll dem einer Standard-Fluidsimulation entsprechen.

Für die Festkörper wird folgendes überprüft:

- Konvergenz von Geschwindigkeit und Kraft
- Anpassung der Geschwindigkeit an den Fluss
- Rotationsrichtung

Die Festkörpergeschwindigkeit muss sich stets an die des umgebenden Flusses anpassen. Dabei müssen Kraft und Geschwindigkeit konvergieren. Da die Rotationsgeschwindigkeit in einem konstanten Fluss nicht konvergiert, da keine bremsenden Kräfte berücksichtigt werden, wird nur die Rotationsrichtung getestet.

LBM-Fluidsimulation

Die Massenerhaltung wurde über die Gleichungen (2.12) und (2.13) geprüft. Da es sich bei der Lattice Boltzmann Methode um ein numerisches Verfahren handelt, sind kleinere Unebenheiten zu erwarten. In Abbildung 4.7 sieht man die Massenerhaltung bei der ungekoppelten LBM-Simulation. Abbildung 4.7 zeigt die Differenz der Massenerhaltung von der Simulation mit und ohne Kopplung. Man kann erkennen, dass während der Beschleunigung des Festkörpers die Massenerhaltung weniger eingehalten wird. Der Wert ist allerdings nur sehr gering und kann daher vernachlässigt werden (Die Masse des Fluids auf dem Gebiet ist ≈ 80000). Sobald der Festkörper mit konstanter Geschwindigkeit mit dem Fluss schwimmt, pendelt die Differenz um den Nullpunkt und sollte sich in etwa ausgleichen.

Das Überprüfen der Impulserhaltung mit den Gleichungen (2.14) und (2.15) funktioniert wie das der Massenerhaltung. Abbildung 4.8 zeigt die Grundlinie der ungekoppelten LBM-Simulation, Abbildung 4.9 die Differenz. Wie auch bei der Massenerhaltung, weichen die Impulserhaltung der gekoppelten Simulation stärker ab, je höher die Beschleunigung des Festkörpers ist. Die Abweichung der Impulserhaltung ist dabei stärker als die der Massenerhaltung und bleibt selbst bei konstanter Geschwindigkeit durchgehend über null. Es gilt aber wieder, dass der Fehler sehr gering und dadurch vernachlässigbar ist.

Die Graphen der Massen- und Impulserhaltung wurden mit einem Gauss-Filter mit Kernelgröße 401, $\sigma = 150$ und $\mu = 0$ geglättet. Dabei ist zu beachten, dass durch die starke Glättung die Werte am Rand des Definitionsbereiches sehr ungenau sein können.

Um überprüfen zu können, ob ein realistisches Strömungsbild entsteht, wurden die Geschwindigkeiten der Strömung mithilfe einer VTK-Ausgabe und Paraview visualisiert (siehe dazu [AGL05]). Abbildung 4.10 zeigt Standbilder der Strömung.

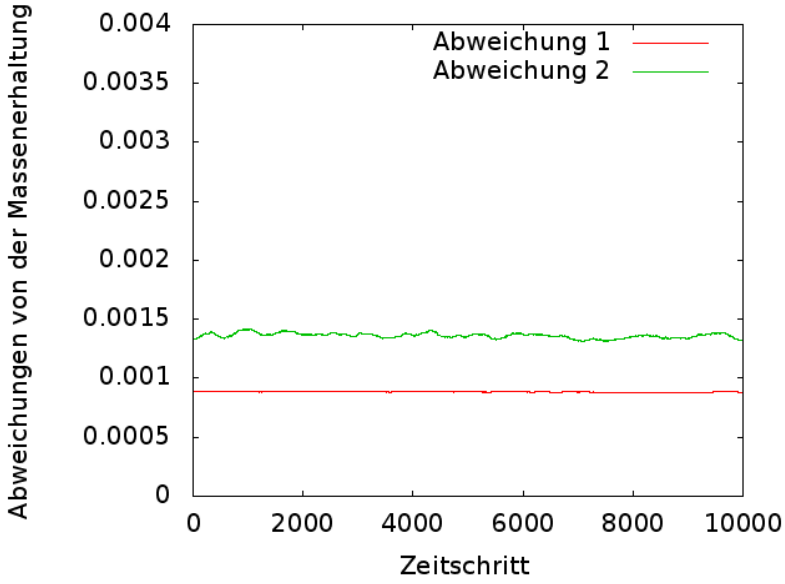


Abbildung 4.6.: Massenerhaltung bei der Standard-LBM-Simulation. „Abweichung 1“ bezieht sich dabei auf die Massenerhaltung der Equilibriumsfunktion aus Gleichung (2.14), „Abweichung 2“ auf die des Kollisionsoperators aus Gleichung (2.15).

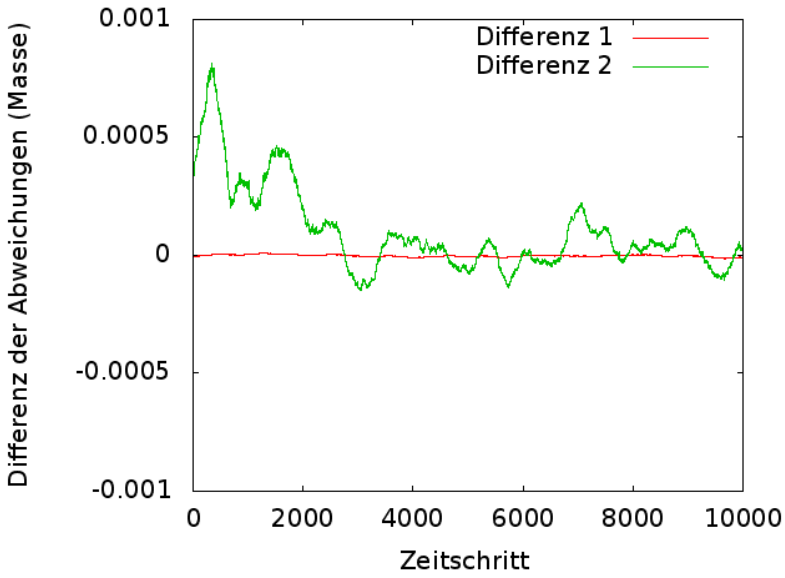


Abbildung 4.7.: Differenz der Massenerhaltungen von gekoppelter und ungekoppelter Simulation. „Differenz 1“ bezieht sich dabei auf die aus Gleichung (2.12), „Differenz 2“ auf die aus (2.13).

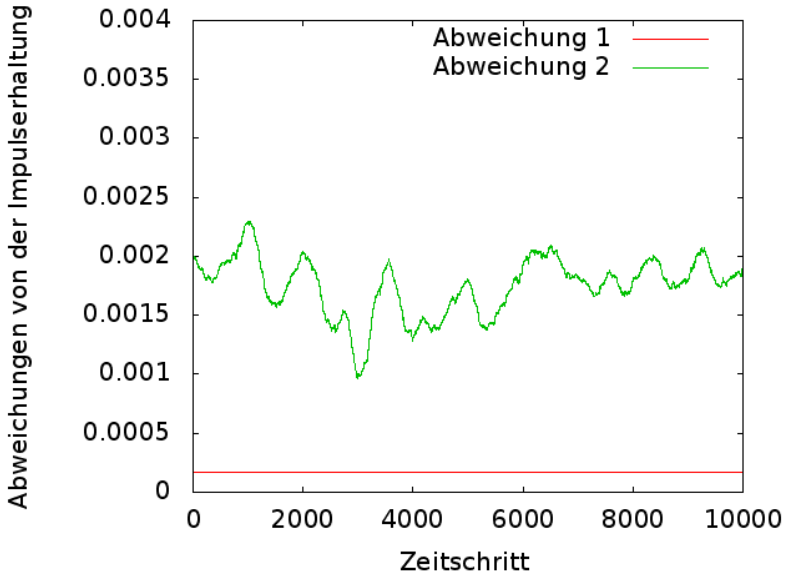


Abbildung 4.8.: Impulserhaltung bei der Standard-LBM-Simulation. „Abweichung 1“ bezieht sich dabei auf die Impulserhaltung der Equilibriumsfunktion aus Gleichung (2.14), „Abweichung 2“ auf die des Kollisionsoperators aus Gleichung (2.15).

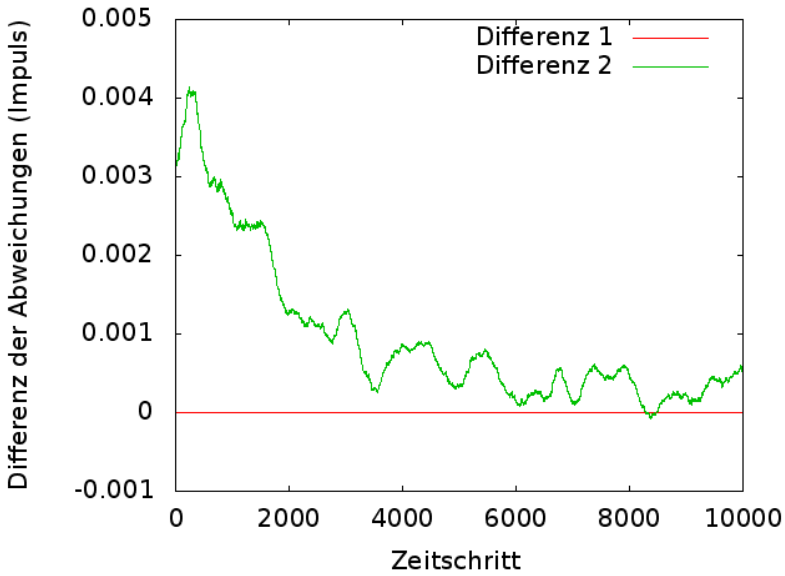


Abbildung 4.9.: Differenz der Impulserhaltungen von gekoppelter und ungekoppelter Simulation. „Differenz 1“ bezieht sich dabei auf die aus Gleichung (2.14), „Differenz 2“ auf die aus (2.15).

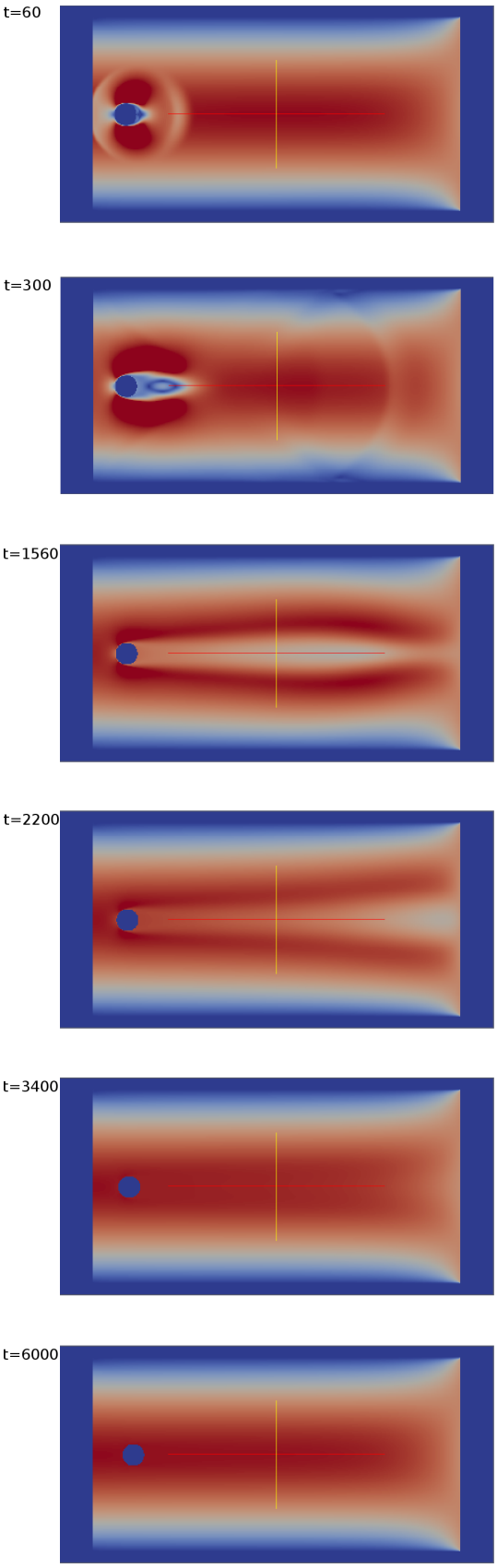


Abbildung 4.10.: Größe der Strömungsgeschwindigkeiten zu verschiedenen Zeitschritten visualisiert mithilfe von VTK und paraview.

$t = 60$ zeigt das Strömungsbild kurz nach dem Einfügen des Festkörpers. Es ist zu erkennen, dass sich langsam das typische Bild entwickelt. Man sieht auch die Schockwelle, die beim Einfügen erzeugt wird und die die Messwerte am Anfang verfälscht.

Bei $t = 300$ ist die Geschwindigkeit des Festkörpers immer noch sehr gering und man sieht das typische Strömungsbild für unbewegte Körper.

Bei $t = 1560$ ist der Festkörper schon schneller. Dadurch werden weniger Fluidpartikel um diesen gelenkt und er treibt etwas Fluid vor sich her.

$t = 2200$ zeigt, dass diese Effekte mit zunehmender Geschwindigkeit des Festkörper auch mehr werden. Bei $t = 3400$ wird dann kaum noch Fluid um den Festkörper gelenkt und bei $t = 6000$ schwimmt der Festkörper letzlich vollständig mit dem Fluss mit.

Dieses Ergebnis entspricht dem erwarteten Verhalten und der Test kann damit als erfolgreich gewertet werden.

Festkörper

Damit der Festkörper nicht unbegrenzt beschleunigt, muss die Kraft auf den Festkörper und damit auch dessen Geschwindigkeit in einem Poiseuille-Fluss konvergieren. Abbildung 4.11 zeigt die Kraft auf den Festkörper über die Zeit für die Kopplungsmethoden, Abbildung 4.12 die Geschwindigkeit. An dieser Stelle reichen jeweils die x-Komponenten, da auch der Fluss nur in x-Richtung fließt.

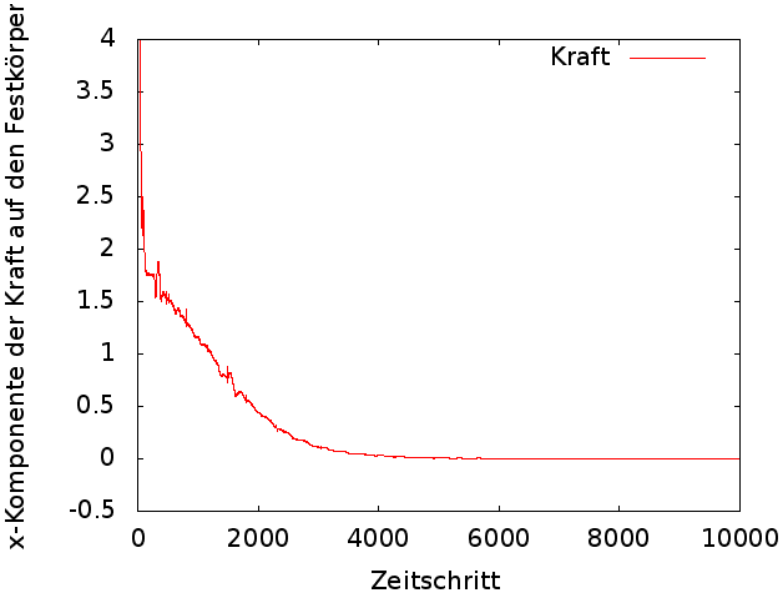


Abbildung 4.11.: x-Komponente der Kraft die auf den Festkörper wirkt abhängig vom Zeitschritt.

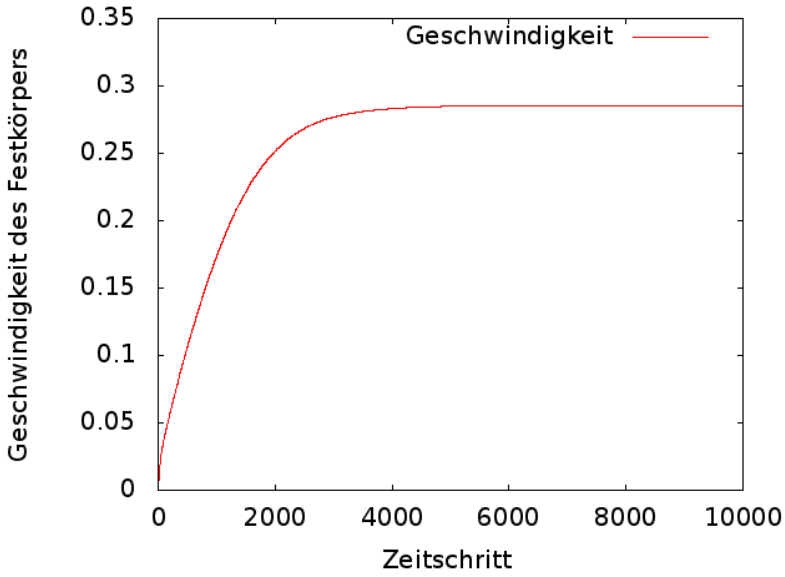


Abbildung 4.12.: x-Komponente der Geschwindigkeit die auf den Festkörper wirkt abhängig vom Zeitschritt.

Die Kraft-Kurve konvergiert gegen 0 und damit die Geschwindigkeitskurve gegen einen festen Wert > 0. Das zeigt, dass die Kopplungsverfahren hinsichtlich der Kraft-/Geschwindigkeitskonvergenz korrekt sind.

Die Geschwindigkeit des Festkörpers in einem Poiseuille-Fluss muss auf der Geschwindigkeitsparabel in dem Bereich sein, in dem sich der Festkörper befindet, da er damit weder schneller noch langsamer als der umgebende Fluss ist. Dies wird in Abbildung 4.13 veranschaulicht.

Abbildung 4.14 zeigt das Ergebnis für die Standard-Kopplungsverfahren.

Die Festkörpergeschwindigkeit konvergiert gegen einen Wert im erwünschten Bereich. Damit sind die Kopplungsverfahren korrekt hinsichtlich der Geschwindigkeit der Körper.

Für den Test der Rotationsrichtung wurde die Startposition der Festkörper aus der Mitte heraus jeweils nach oben bzw. nach unten versetzt. Dadurch ist der umgebende Fluss an einer Seite des Festkörpers schneller. Wird die Startposition nach oben verschoben, ist der Fluss an der Unterseite schneller und es wird eine Rotation gegen den Uhrzeigersinn erwartet, analog dazu erwartet man bei der Verschiebung nach unten eine Rotation im Uhrzeigersinn. Diesen Aufbau zeigt Abbildung 4.15.

Das Ergebnis für den nach unten verschobenen Festkörper zeigt Abbildung 4.16, für den nach oben verschobenen Abbildung 4.17. Bei dem nach unten verschobenen Festkörper ist die Winkelbeschleunigung positiv. Das bedeutet, dass der Festkörper sich im Uhrzeugersinn dreht. Der nach oben verschobene Festkörper beschleunigt wie erhofft gegen den Uhrzeigersinn. Das zeigt, dass unsere Rotationsrichtung korrekt ist. Man sieht auch die Konvergenz gegen ein Werteintervall dessen Rän-

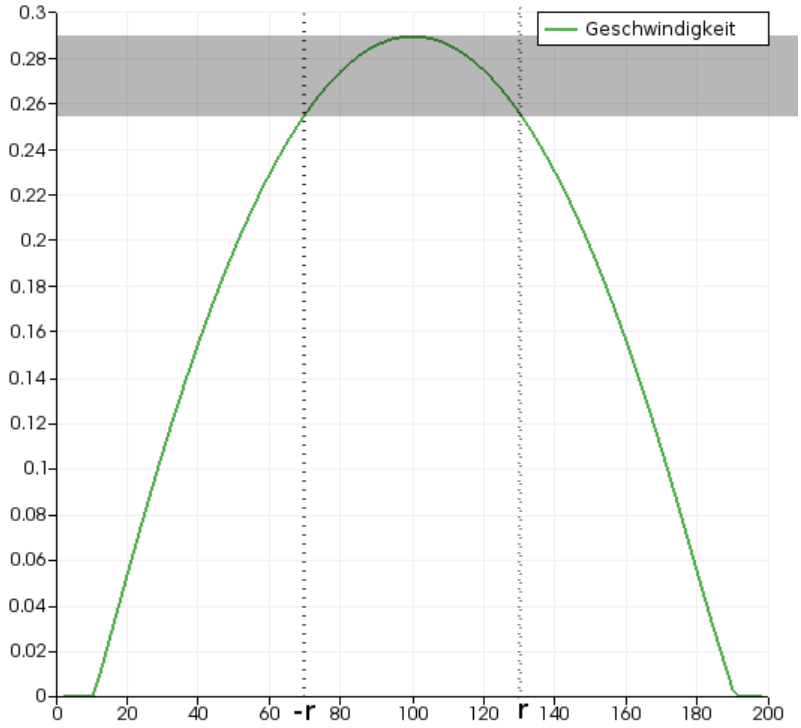


Abbildung 4.13.: Gültige Festkörpergeschwindigkeit in der Parabel des Poiseuille-Flusses. Der Bereich in dem die Geschwindigkeit liegen muss, ist grau markiert. $-r$ und r begrenzen die Lage des Festkörpers.

der beide > 0 oder beide < 0 sind. Das führt zu der angesprochenen, potentiell endlos steigenden Winkelgeschwindigkeit.

4.2.3. Ergebnis

Die Verifikation der Kopplungsverfahren war in allen getesteten Punkten erfolgreich. Es wurden zwar nur die Graphen der als Standard festgelegten Verfahren gezeigt, jedoch wurden die der anderen Verfahren auch getestet. Dabei waren diese stets korrekt. Die Unterschiede werden im folgenden Abschnitt erläutert.

4.2.4. Vergleich der Verfahren

Der Vergleich der Verfahren wurde anhand mehrerer Kriterien durchgeführt. Dabei wurde sich in dieser Arbeit auf die visuell erkennbaren Unterschiede beschränkt und nicht etwa der Fehler analytisch bestimmt. Die Kriterien sind:

- Kraft/Geschwindigkeit des Festkörpers

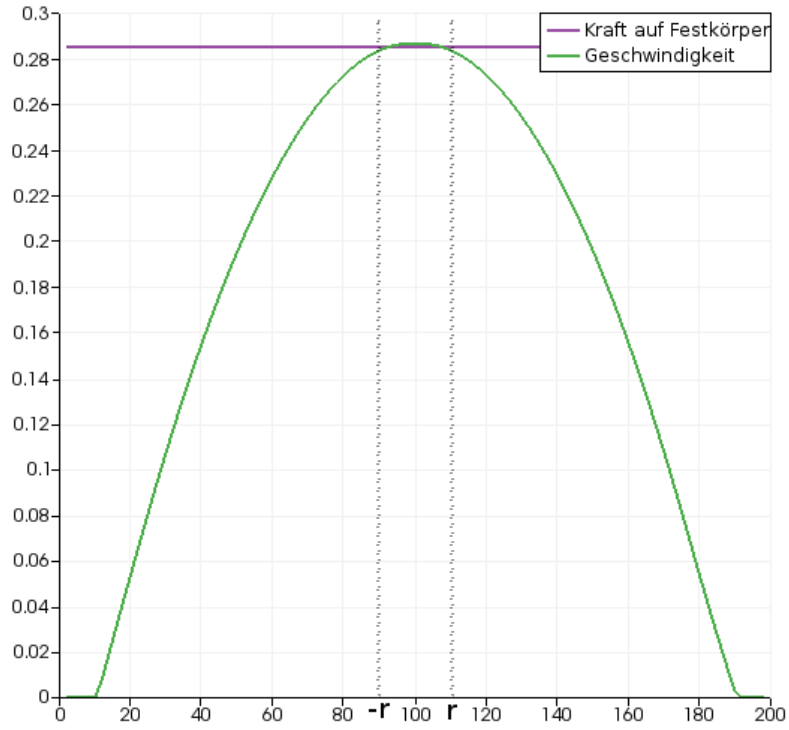


Abbildung 4.14.: Geschwindigkeit des Festkörpers im Poiseuille-Fluss. $-r$ und r begrenzen die Lage des Festkörpers.

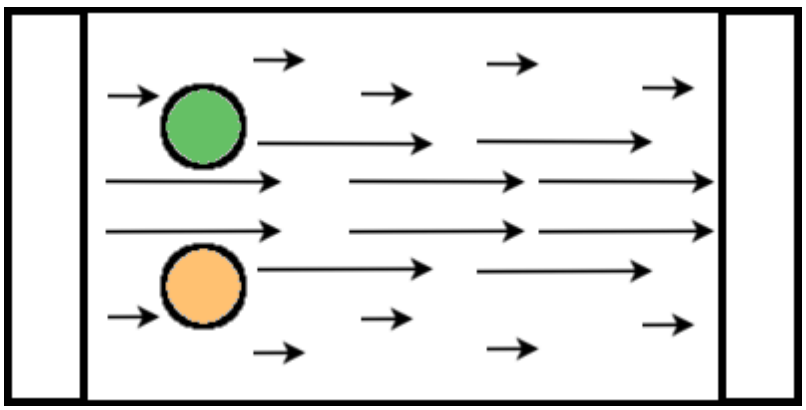


Abbildung 4.15.: Skizze des Aufbaus zum Testen der Rotationsrichtung. Der nach oben verschobene Festkörper ist grün, der nach unten verschobene orange.

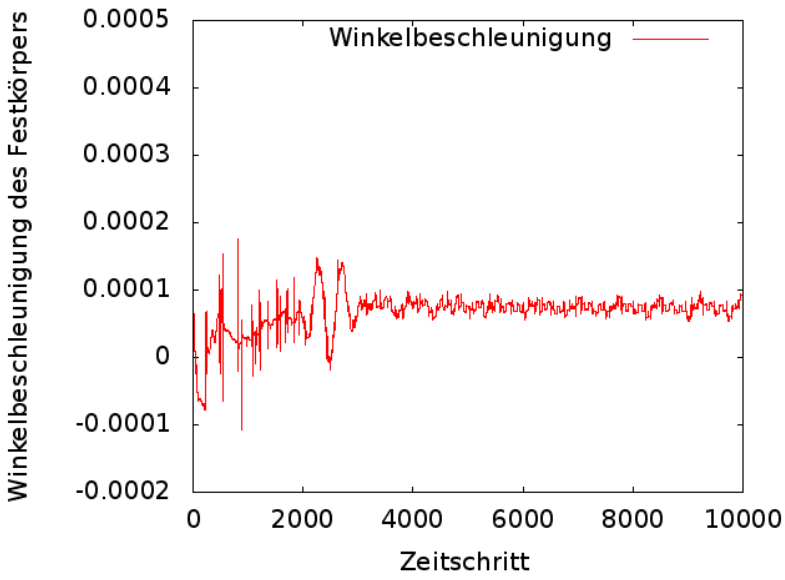


Abbildung 4.16.: Winkelbeschleunigung des nach unten verschobenen Festkörpers abhängig vom Zeitschritt. Positive Werte bedeuten, dass eine Beschleunigung mit dem Uhrzeigersinn stattfindet.

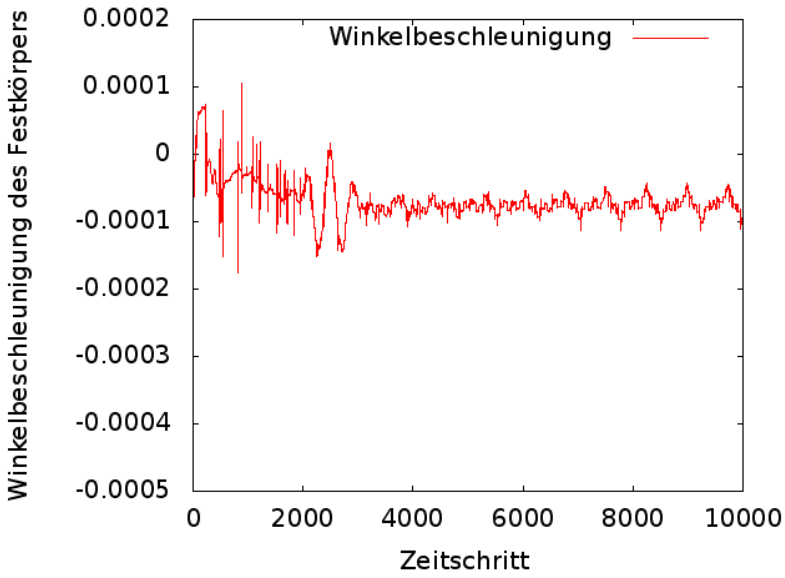


Abbildung 4.17.: Winkelbeschleunigung des nach oben verschobenen Festkörpers abhängig vom Zeitschritt. Negative Werte bedeuten, dass eine Beschleunigung gegen den Uhrzeigersinn stattfindet.

- Strömungsbild
- Laufzeit

Für das Messen der Laufzeit wurde für einen Durchlauf des Programms die durchschnittliche Zeit zwischen den Frames gemessen und daraus die durchschnittlichen FPS berechnet. Da Talfer nicht mehr als 60 FPS ausgibt, wurde die Anzahl der LBM-Zeitschritte pro Durchlauf auf 4 erhöht. Um den Einfluss der Festkörper zu vergrößern, wurde zusätzlich noch der Radius auf 40 gesetzt. Damit die Simulation stabil bleibt, wurde die Startposition des Festkörpers zu $(150, 100)^T$ geändert.

Mehrere Methoden wurden dabei für die Randbedingen und die Initialisierung von Zellen vorgestellt.

Die Randbedingungen

Zum Vergleich stehen die Randbedingungen aus [ITR08] (Gleichung (3.2)) und die aus [BFL01] (Gleichung (3.1)). Abbildung 4.18 zeigt die Differenz der Kraft auf den Festkörper. An dieser erkennt man leichte Unterschiede in der berechneten Kraft. Betrachtet man zum Vergleich die Kraft-Kurve in Abbildung 4.11, sieht man, dass die Größe der Abweichung proportional zur Größe der Kraft ist. Dadurch ist diese Abweichung zu jedem Zeitpunkt relativ gering. Das erkennt man an der Differenz der Geschwindigkeiten in Abbildung 4.19. Trotz leichter Unterschiede bei der Beschleunigung des Festkörpers konvergieren beide Geschwindigkeiten auf ungefähr denselben Wert. Die Strömungsbilder der Verfahren sind nahezu identisch. Dazu zeigt Abbildung 4.20 das Strömungsbild der Bouzidi-Methode zum Zeitpunkt $t = 1560$ zum Vergleich mit Abbildung 4.10.

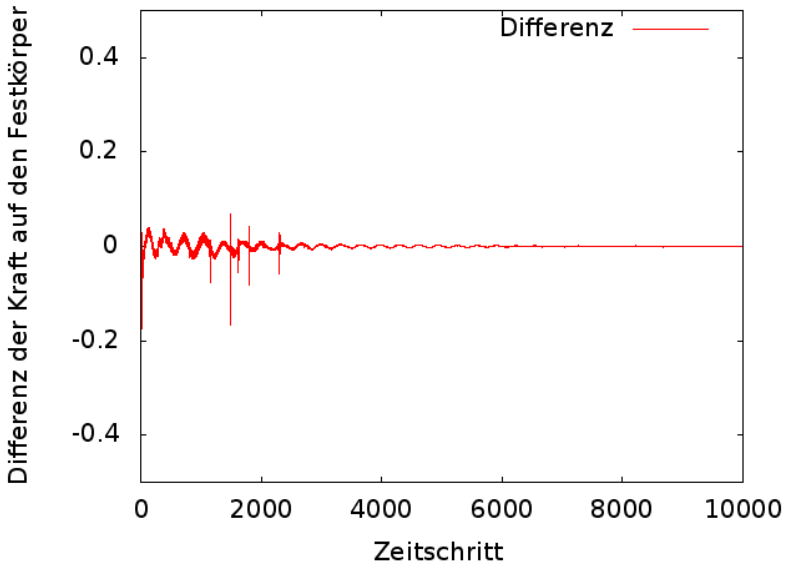


Abbildung 4.18.: Differenz der Kräfte von der Bouzidi- und der Iglberger-Variante.

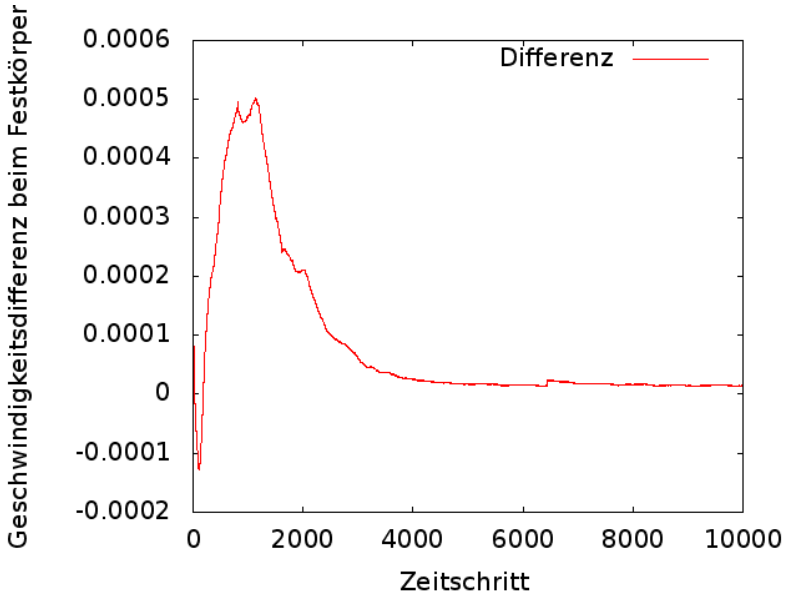


Abbildung 4.19.: Differenz der Geschwindigkeiten von der Bouzidi- und der Iglberger-Variante. Positive Werte bedeuten eine höhere Geschwindigkeit bei der Bouzidi-Methode.

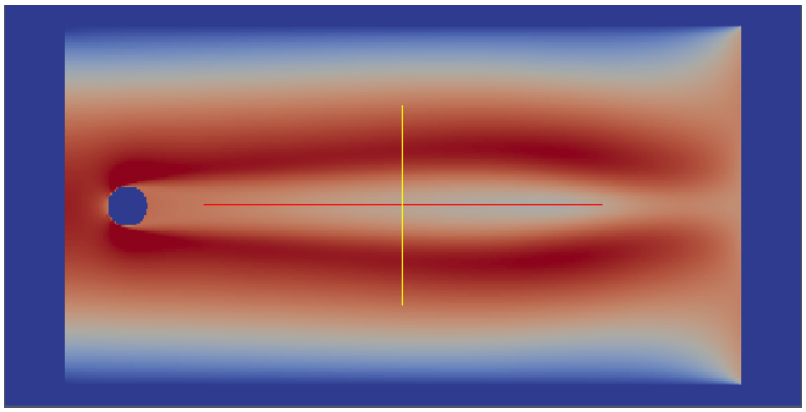


Abbildung 4.20.: Strömungsbild der Simulation mit den Bouzidi-Randbedingungen zum Zeitschritt $t = 1560$.

Das Ergebnis für die Laufzeiten zeigt Abbildung 4.21. Ein Simulationsdurchlauf ist dabei ein Durchgang der Pipeline. Die Iglberger-Randbedingungen haben hier überwiegend die bessere Laufzeit.

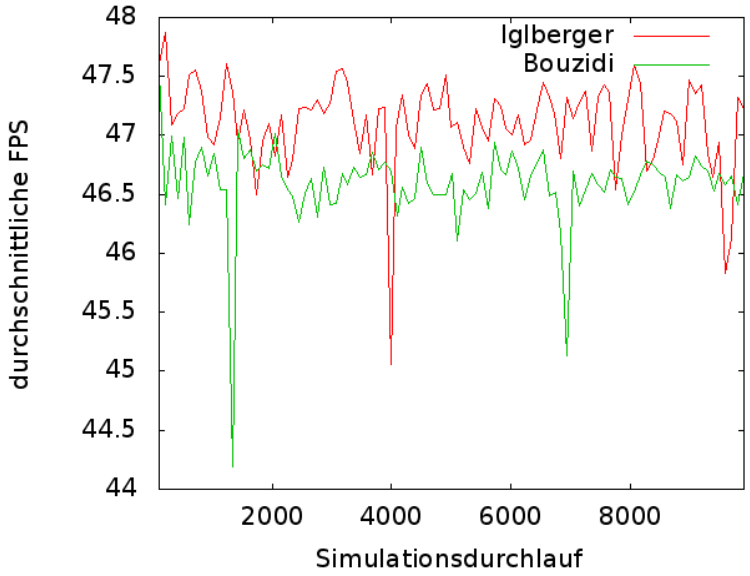


Abbildung 4.21.: Laufzeitvergleich der Randbedingungen aus [ITR08] und [BFL01]. Der Durchschnitt der FPS ist über 100 Simulationsdurchläufe.

Initialisierungsverfahren für Zellen

Die verschiedenen Initialisierungsverfahren sind die Standardinitialisierung, die Equilibriumsinitialisierung und die Non-Equilibriumsinitialisierung. Ähnlich wie bei den Randbedingungen haben die Equilibriums- und die Non-Equilibriumsinitialisierung keine visuell erkennbaren Unterschiede bei Kraft/Geschwindigkeit bzw. Strömungsbild. Auf die Präsentation von Graphen und Bildern wird hierbei verzichtet.

Bei der Standardinitialisierung gibt es deutlich sichtbare Unterschiede. Diese entstehen aufgrund der Tatsache, dass Zellen bewusst falsch (aber stabil) initialisiert werden und sich an den Fluss anpassen müssen. Bei der auf den Festkörper wirkenden Kraft entstehen dadurch Peaks (siehe Abbildung 4.22) und bei dem Strömungbild Artefakte hinter dem Festkörper (Abbildung 4.24). Die Auswirkungen auf die Geschwindigkeit sieht man in Abbildung 4.23. Die Differenz zur Equilibriumsinitialisierung weist ein ähnliches Zackenmuster auf wie die Kraft-Kurve. Dieses entsteht, da Fluidgitterpunkte nur in einigen Zeitschritten reaktiviert werden und die Simulation dazwischen keine Initialisierung vornehmen muss.

Die Laufzeiten sieht man in Abbildung 4.25. Wie erwartet hat die Standardinitialisierung aufgrund ihrer Einfachheit die beste Laufzeit. Die Non-Equilibriumsinitialisierung benötigt mehr Zeit als die Equilibriumsinitialisierung, weil zusätzlich noch der Non-Equilibriumsteil berechnet werden muss.

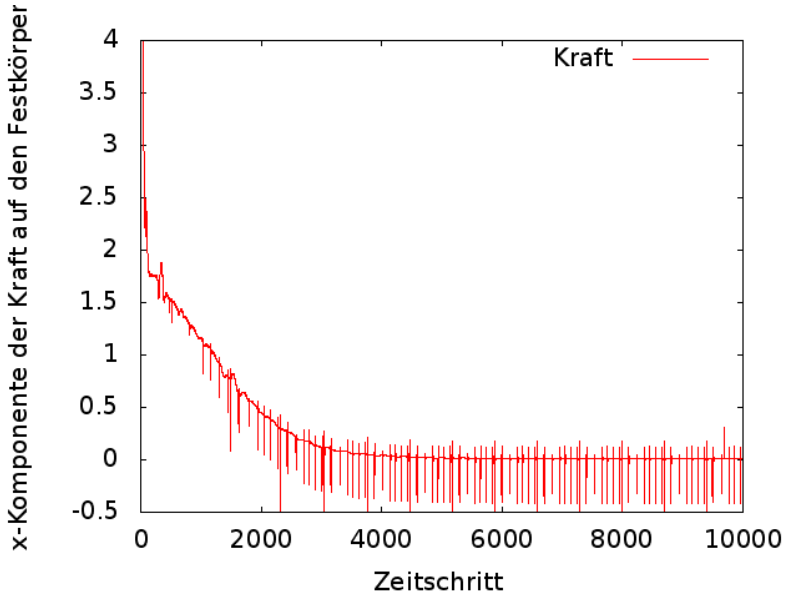


Abbildung 4.22.: Kraftkurve bei der Verwendung der Standardinitialisierung.

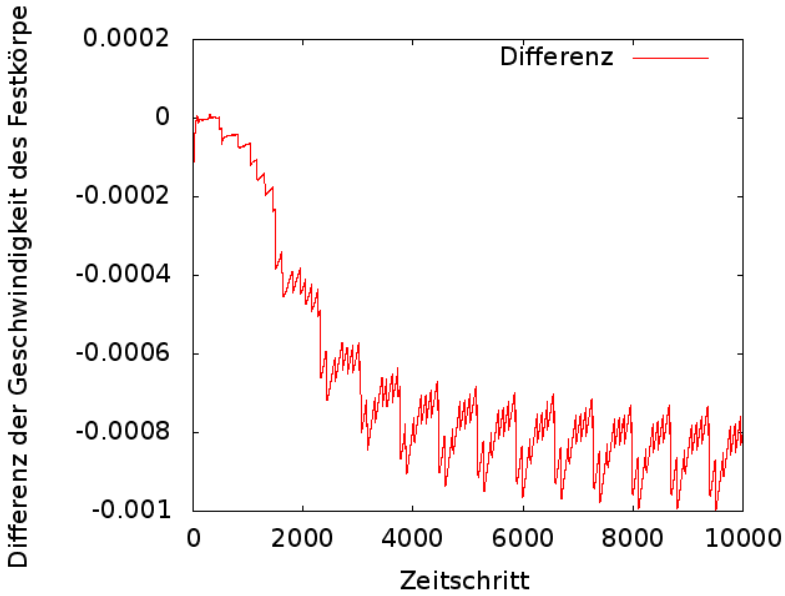


Abbildung 4.23.: Geschwindigkeitsdifferenz von der Verwendung der Standardinitialisierung und der des Standardverfahrens.

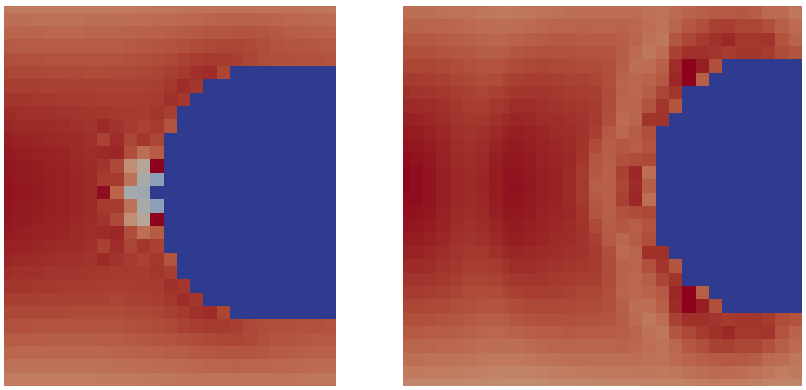


Abbildung 4.24.: Zoom auf das Strömungsbild der Simulation mit der Standardinitialisierung zum Zeitschritt $t = 4440$ (links) und $t = 4480$ (rechts). Dabei reicht die Farbskala hier von 0.26-0.29 (statt 0-0.29). Hinter dem Festkörper ist deutlich ein Artefakt zu erkennen.

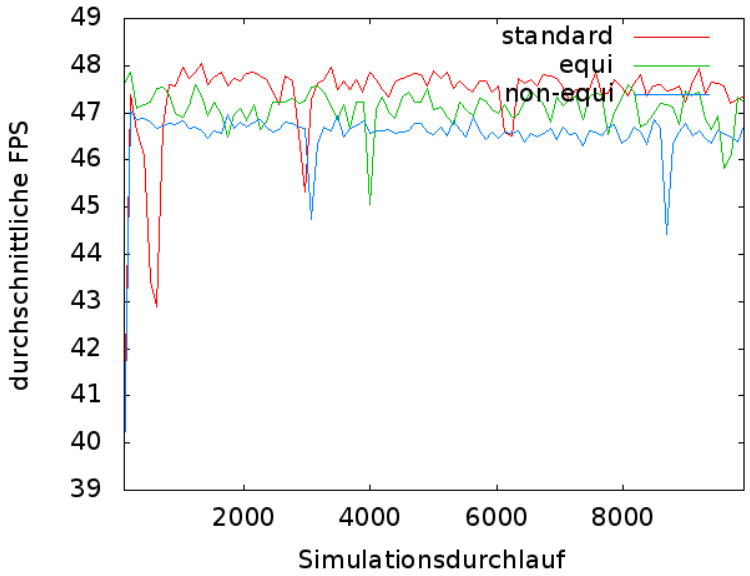


Abbildung 4.25.: Laufzeitvergleich der Initialisierungsverfahren

4.2.5. Laufzeitanalyse

Hier wird untersucht, wie sich die Laufzeit zur Größe des Festkörpers verhält. Das Testverfahren wurde in Abschnitt 4.2.4 beschrieben. Um die Auswirkung des Festkörperradius zu untersuchen, wurde dieser beim Laufzeit-Testverfahren in jedem Durchlauf schrittweise verkleinert. Das Ergebnis wird in Abbildung 4.26 veranschaulicht. Man sieht, dass der Radius des Festkörper kaum Auswirkungen auf

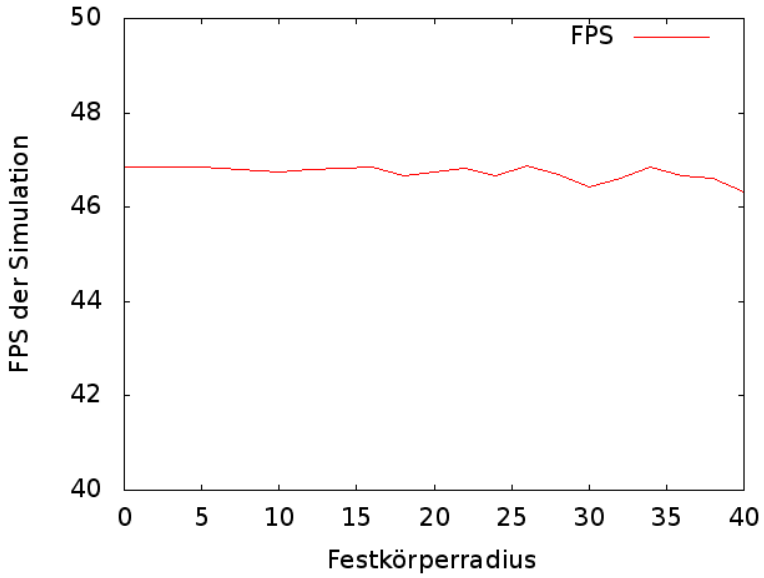


Abbildung 4.26.: FPS der Simulation in Abhängigkeit vom Radius des Festkörpers.

die Laufzeit hat. Das liegt daran, dass bei Erhöhung des Radius nicht nur die Festkörpergitterzellen an der Fluidgrenze mehr werden, sondern auch die inaktiven Zellen. Der reine LBM-Teil der Simulation kann also schneller berechnet werden und gleicht sich hier mit den Mehrkosten für den Kopplungsteil aus. Damit schränkt die Kopplung mit dem Festkörper die Laufzeit der Simulation nicht ein.

5. Fazit und Ausblick

In dieser Arbeit wurden Kopplungsmethoden für die Kopplung der LBM-Simulation mit einer Festkörper-Engine vorgestellt. Diese mussten zuerst verifiziert werden. Dabei wurden sowohl der Einfluss von den Festkörpern auf das Fluid, als auch der vom Fluid auf die Festkörpern überprüft. Beide lieferten zufriedenstellende Ergebnisse, wodurch die Kopplung der Festkörper-Engine mit der LBM-Simulation als Erfolg gewertet wurde. Ein Vergleich der Verfahren hat gezeigt, dass visuell keine großen Unterschiede wahrzunehmen sind, solange man Methoden wählt, die die vorliegende Physik ausreichend genau approximieren. So hatte die Standardinitialisierung negative Auswirkungen auf das Strömungsbild. Eine Echtzeitfähigkeit wurde dabei erreicht, nämlich etwa dieselbe wie die der ungekoppelten Simulation. Das lag daran, dass sich die Mehrkosten der Kopplung mit den Einsparungen durch die Deaktivierung von Fluidpunkten unabhängig von der Festkörpergröße ausgeglichen hat.

Für die Anwendung in einem interaktiven Strömungslöser oder Spiel kam allerdings ein Problem auf. Die Geschwindigkeiten, bei denen die Simulation stabil läuft, sind so gering, dass die Festkörper sich optisch nur sehr langsam fortbewegen. Dadurch ist eine Interaktion nur schwer umzusetzen bzw. vom Benutzer nachvollziehbar. Eine künstliche Skalierung der Festkörpergeschwindigkeit in der Festkörper-Engine würde wiederum die physikalische Korrektheit außer Kraft setzen.

Ausblick

Im Hinblick auf die Umsetzung als interaktive Strömungsimulation können noch viele Erweiterungen durchgeführt werden. So braucht man z.B. die Möglichkeit externe Kräfte miteinzubeziehen. Diese müssen allerdings richtig verarbeitet werden, damit sie die Stabilität der LBM-Simulation nicht beeinträchtigen. Zusätzlich können noch weitere Arten der Interaktion hinzugefügt werden, mit denen der Benutzer ausgewählte Parameter der Simulation, wie z.B. die maximale Einflussgeschwindigkeit, beeinflussen kann.

Legt man Wert auf die physikalische Korrektheit, kann man die verschiedenen Verfahren statt nur optisch auch analytisch vergleichen. Einen Ansatz dazu liefert [Cai08]. Dabei werden die Ergebnisse der LBM-Simulation mit denen aus den Navier-Stokes-Gleichungen verglichen. Dadurch erhält man einen analytischen Fehler, mit welchem man die Verfahren echt ordnen kann. Außerdem wird in dieser Arbeit die Verifikation nur für ein festgesetztes Sample durchgeführt. Besser wäre ein Verfahren, das Korrektheit für ein bestimmtes Intervall beweist.

A. Algorithmen

Hier werden Algorithmen für die LBM-Simulation und die Festkörper-Engine vorgestellt.

Algorithmus 1 Festkörper-Engine

```
1: initialiseRigidBodies()
2: while running do
3:     recieveData(F_p, τ_p)
4:     for all RigidBody P do
5:         updateRigidBody(P, F_p[P], τ_p[P])
6:     end for
7:     sendData(RigidBodyData)
8: end while
```

Algorithmus 2 LBM-Algorithmus für einen LBM-Zeitschritt pro Festkörper-Zeitschritt

Symbols:
Kraft auf die Festkörper: F_p
Drehmoment auf die Festkörper: τ_p

initialiseLBM()
while running **do**
  recieveData(RigidBodyData)
  F_p,$\tau_p = \{0, 0, ...\}$
  // *iteriere über alle Gitterpunkte*
  for $y = 0$ **to** height $- 1$ **do**
    for $x = 0$ **to** width $- 1$ **do**
      // *Kollisionsschritt*
      if isFluidNode(x, y) **then**
        if isCellReactivated(x, y) **then**
          initialiseCell(x, y)
        end if
        ρ =computeRho(f_i)
        $\mathbf{u}$ =computeVelocity(f_i, $\mathbf{c}_i$, ρ)
        f_i^{eq} =computeEquilibriumDist(ρ, $\mathbf{u}$)
        f_i^* =computeCollision(f_i, f_i^{eq}, τ)
      end if
      // *Strömungsschritt*
      if isFluidNode(x, y) **then**
        // *in alle Richtungen des D2Q9-Modells*
        for $i = 0$ **to** 8 **do**
          if isFluidNode($(x, y) + \mathbf{c}_i$) **then**
            stream(f_i^*)
          end if
          if isRigidBody($(x, y) + \mathbf{c}_i$) **then**
            f_{ref} =computeMovingBoundaryConditions()
            F_p[getRigidBodyIndex($(x, y) + \mathbf{c}_i$)] = computeForce(f_{ref})
            τ_p[getRigidBodyIndex($(x, y) + \mathbf{c}_i$)] = computeTorque(f_{ref})
            stream(f_{ref})
          end if
          if isWall($(x, y) + \mathbf{c}_i$) **then**
            f_{ref} =computeStationaryBoundaryConditions()
            stream(f_{ref})
          end if
        end for
      end if
      if isInflow(x, y) or isOutFlow(x, y) **then**
        streamIntoFluidDomain()
      end if
    end for
  end for
  sendData(F_p, τ_p)
end while

Literaturverzeichnis

[AGL05] J. Ahrens, B. Geveci, C. Law. 36 ParaView: An End-User Tool for Large-Data Visualization. *The Visualization Handbook*, S. 717, 2005. (Zitiert auf Seite 36)

[BFL01] M. Bouzidi, M. Firdaouss, P. Lallemand. Momentum transfer of a Boltzmann-lattice fluid with boundaries. *Physics of Fluids (1994-present)*, 13(11):3452–3459, 2001. (Zitiert auf den Seiten 7, 22, 45 und 47)

[Cai08] A. Caiazzo. Analysis of lattice Boltzmann nodes initialisation in moving boundary problems. *Progress in Computational Fluid Dynamics, An International Journal*, 8(1):3–10, 2008. (Zitiert auf den Seiten 25 und 51)

[Con88] P. Constantin. *Navier-stokes equations*. University of Chicago Press, 1988. (Zitiert auf Seite 9)

[ITR08] K. Iglberger, N. Thürey, U. Rüde. Simulation of moving particles in 3D with the Lattice Boltzmann method. *Computers & Mathematics with Applications*, 55(7):1461–1468, 2008. doi:10.1016/j.camwa.2007.08.022. URL http://linkinghub.elsevier.com/retrieve/pii/S0898122107006256. (Zitiert auf den Seiten 7, 22, 26, 34, 45 und 47)

[Lad94] A. J. Ladd. Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 2. Numerical results. *Journal of Fluid Mechanics*, 271:311–339, 1994. (Zitiert auf Seite 12)

[Mon92] J. J. Monaghan. Smoothed particle hydrodynamics. *Annual review of astronomy and astrophysics*, 30:543–574, 1992. (Zitiert auf Seite 10)

[Neu13] P. Neumann. *Hybrid Multiscale Simulation ApApproach for Micro- and Nanoflows*. Dissertation, Technische Universität München, 2013. (Zitiert auf den Seiten 15, 24 und 27)

[Pfi12] S. Pfister. *Grundlagen für die Anwendung numerischer Strömungssimulation auf Brandszenarien in Industrieanlagen*. Dissertation, Universitätsbibliothek der Technischen Universität Berlin, 2012. (Zitiert auf Seite 9)

[Rap04] D. C. Rapaport. *The art of molecular dynamics simulation*. Cambridge university press, 2004. (Zitiert auf Seite 9)

[Sch08] U. D. Schiller. *Thermal fluctuations and boundary conditions in the lattice Boltzmann method*. Dissertation, Johannes Gutenberg-Universität, Mainz, 2008. (Zitiert auf Seite 13)

[ST96] M. Schäfer, S. Turek. Benchmark Computations of Laminar Flow Around a Cylinder. *Notes on Numerical Fluid Mechanics*, 52:547–566, 1996. (Zitiert auf Seite 17)

[Suc01] S. Succi. *The lattice Boltzmann equation: for fluid dynamics and beyond.* 2001. (Zitiert auf den Seiten 14 und 17)

[WG00] D. Wolf-Gladrow. *Lattice-gas cellular automata and lattice Boltzmann models: An Introduction.* 2000. URL http://books.google.com/books?hl=en&lr=&id=cHcpWgxAUu8C&oi=fnd&pg=PA1&dq=Lattice-Gas+Cellular+Automata+and+Lattice+Boltzmann+Models+-+An+Introduction&ots=I9j6LqrVAd&sig=rdq3wsdlKbW2RIwxmbyQ93W6Odc. (Zitiert auf Seite 14)

[WZHW94] M. Wittmann, T. Zeiser, G. Hager, G. Wellein. Comparison of different propagation steps for lattice Boltzmann methods. *Computers & Mathematics with Applications*, 65:924–935, 1994. (Zitiert auf Seite 12)

Alle URLs wurden zuletzt am 07. 06. 2015 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift